Information Society Technologies

PReVENT

Preventive and Active Safety Applications
Integrated Project
Contract number FP6-507075
eSafety for road and air transport

Subproject

Deliverable type: IP Deliverable

Deliverable name:

D12.42

Interface and Data Entity Specifications (Final)

Part 3 – API Specification

Version number	10
Lead contractor	NAVIGON AG
Confidentiality status	Public
Due date of deliverable	10.12.2005
Actual submission date	15.03.2006
File name	PR-12420-IPD-051130-v10-NAV-Specification_Part3_API

Authors

Maria Perchina, NAVIGON AG
Maxim Arbitmann, Continental Teves GmbH
Johannes Angenvoort, NAVIGON AG
Axel Varchmin, Blaupunkt GmbH
Sinisa Djurekovic, NavTeq B.V.

SubProject Leader

Vincent Blervaque
ERTICO – ITS Europe
Avenue Louise, 326
1050 Brussels / Belgium

Phone +32 2 400 07 24
Fax +32 2 400 07 01
E-mail v.blervaque@mail.ertico.com

Project Co-ordinator

Matthias M. Schulze
DaimlerChrysler AG
HPC T729
70546 Stuttgart / Germany

Phone +49 711 17 41888
Fax +49 711 17 41242
E-mail matthias.m.schulze@daimlerchrysler.com

Revision chart and history log

Version	Date	Reason
0.1	17.03.2005	Memory management reworked, LINK_Id and PATH_ID are introduced in functions; „convenience" functions are deleted.
0.2	23.06.2005	Added definition of Path for Path Reconstructor API
0.3	27.01.2006	Document is reworked to be used together with specification document created by Doxygen program (Appendix B). Appendix A specifying traffic signs is added
0.9	21.02.2006	Finalisation for Peer-Review
1.0	13.03.2006	Peer Review Comments

Technical Abstract

The MAPS&ADAS subproject within the EC PReVENT Project seeks to develop and validate both an applicable standard for the collection, maintenance and provision of safety content enhanced digital map databases to be used in advanced driver assistance systems (ADAS) and Navigation Applications, as well as a standard interface from navigation systems or general positioning and map systems towards ADAS that make use of map data (e.g., for track preview purposes).

Within the PReVENT WP12.420 work package the specification of the interface and data entity specifications necessary to build and access the ADAS horizon is carried out.

The deliverable D12.42 contains these specifications in a final version, considering those parts of the specification that were validated within the project. For the sake of clearness it is split into three main parts.

This deliverable D12.42 Part 3 describes the application programming interface (API), which is provided to a customer as a standardized ANSI C library. Appendix B contains the API design specification, it can be seen as an own sub-document.

In this preliminary version only those functions and concepts are specified that are validated within MAPS&ADAS.

Table of contents

List of figures

List of tables

1. Introduction

This document contains the specification of the API that allows control of and access to the ADAS Horizon. Using this API, any ADAS application is able to connect to the ADAS Horizon Reconstructor and access data by using unified methods provided for this application.

Based on the functionality, all definitions and functions are separated into four groups called *Controller, Positioner, PathReconstructor* and *CurveModeller.* Every group is considered (and named) as a *Component.*

In order to separate API names from names used by an application, the common prefix AH (abbreviation from ADAS Horizon) is used. This prefix is declared in all name declarations (for types, enumeration, functions, etc.) but can be skipped in a detailed description placed in the text.

Chapters 2 and 3 are used to explain approaches and common principles of the system design and to provide the overview descriptions of components. The detailed API specification created by Doxygen tool is placed in the Appendix B of this document.

2. Common types and definitions

This chapter describes the declaration and initialization of variables and types most commonly used in the Horizon. Some of them are exactly the same as those defined in ANSI C, while others are unique to the Horizon (own “derived” types, which are based on types already defined). These declarations are used in order to construct shorter names for types. Others also encapsulate implementation details that may be changed at a later stage.

2.1. Basic types in C language

Basic types in C language are *character*, *integer* and *floating point. Integer* type comes in three sizes, *short*, *int* and *long* and can be used with modifiers *signed* or *unsigned*. Modifier *signed* is usually the default and not required.

Exact size and capacity of a type varies with the machine and operating system. Description of C types and required capacities are given in Table 1.

Table 1. Fundamental Types of the C Language.

Type	Size	Capacity	Contents
char	1 byte	-128 to +127 or 0 to 255	Type **char** contains members of the execution character set... Type can be used with *signed* and *unsigned* modifiers.
short	2 bytes	-32,768 to +32,767 or 0 to 65,535	Type **short int** (or simply **short**) is larger than or equal to the size of type **char**, and shorter than or equal to the size of type **int**. Type can be used with *signed* and *unsigned* modifiers.
int	2 or 4 bytes		Type **int** is larger than or equal to the size of type **short int**, and shorter than or equal to the size of type **long**. Type can be used with *signed* and *unsigned* modifiers.
long	4 bytes	-2,147,483,648 to +2,147,483,647 or 0 to 4,294,967,295	Type **long** (or **long int**) is larger than or equal to the size of type **int**. Type can be used with *signed* and *unsigned* modifiers.
float	4 bytes	3.4E+/-38 (7 digits)	Type **float** is the smallest floating type.
double	8 bytes	1.7E+/-308 (15 digits)	Type **double** is a floating type that is larger than or equal to type **float.**

2.2. ADAS Horizon types

Based on a type content and structure, all types in ADAS Horizon are separated into three groups: *C basic types and short-named types, content-named types* and *complex user types*.

The set of **Basic types** consists of the basic C types. In order to unify names, redefinitions (typedefs) are used to declare basic types in the system. The types included into the **Short-named types** set are redefinitions (for convenience) of C basic types. These types are *AH_UINT16 (unsigned short*), *AH_UINT32* (*unsigned int*) and so on.

The **Content-named types** are derived types based on the Basic types. They are declared in order to provide a more native (for example, *AH_COORDINATE: AH_INT32*) and realization-independent way to define main terms and definitions used in the system. Notice that for representation of *float* values one of *integer* types is used. Thus, values should be divided by the resolution if necessary.

Detailed specification for types mentioned above is given in Appendix B.

Last, the **Complex user types** set contains enumerations and the composed (like structures) types used in the system. For example, the composite type *AH_GEOPOINT* is a structure, which has two elements; *latitude* and *longitude*, both are *AH_COORDINATE* type.

2.3. Definitions

Float values in the system are represented as integer. All resolutions used for converting are defined as well as associated values ranges are named relative to the type name (for example, for the type AH_HEADING the name AH_HEADING_RES is used for the resolution and AH_HEADING_MIN and AH_HEADING_MAX defines the range of possible values). Detailed descriptions are grouped in **definition** sub-sections of type specification sections in Appendix B.

In addition to the definitions mentioned above, the Horizon uses a unified set of **#defines** containing the definition for commonly used (like *boolean true* and *false*) and undefined values.

2.4. Error handling

All functions of the API returns standardized success code. Negative return values indicates errors, 0 indicates success, while positive values are warnings.

Note, that the current document does not include a specification for ADAS CAN-bus interface errors like CAN connection and (or) transmission failures, errors in parsing of CAN messages, etc.

3. Memory management issues

Dependent on the number of applications and purposes, volume of available memory and other reasons, the architecture of Horizon Provider, Horizon Reconstructor and their interaction with applications can be different.

Several architecture schemes can be proposed. For example, in case a user has only one application and restricted memory volume, the Horizon can be realized in form of static or dynamic libraries, so the Reconstructor runs as part of the common process (see Fig. 1):

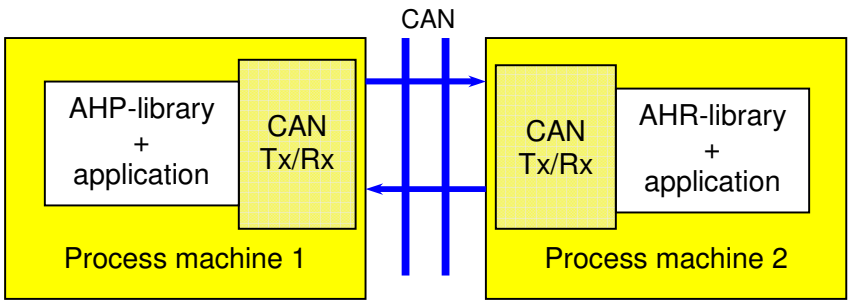

Fig. 1. Single process for the ADAS Horizon Reconstructor and application.

Alternatively, the ADAS Horizon can run in a separated process, which can be used connected with one or several application processes.

API 1.0 is developed to be used as the simple scheme, where an application uses an ADAS Horizon library, so that both the ADAS Horizon Reconstructor (AHR) and the application (Application) run in the same process.

Main design principles for the API 1.0 are introduced below.

1. ADAS Horizon is developed as multithreaded process, where two threads are running. The first one, "receiving thread" (CAN-Receiver/Transmitter) handles CAN communication, and the second handles AHR together with the Application. Details of CAN communication realization are not considered here.
2. In order to minimize memory usage, some routines (accessing configuration parameters) use pointers to structures and variables. This approach requires that data should exist for the application's purposes and cannot be destroyed or updated by the ADAS Horizon while they are requested. At the same time in other routines pointers to objects allocated in memory by an application are used. All API structures can be separated into two groups.

Group one consists of constant objects like metadata. The content of these objects cannot be changed in runtime; access to these structures is done by using const pointers.

Group two consists of objects that are changeable by Horizon updating. Application must allocate memory and provide a pointer to this memory to the API. API fills that memory performing conversions and filtering of the data stored in the ADAS Data Store. Some API functions operate with arrays of structures. This case is more complicated, since the application should know the number of elements in an array in order to allocate memory properly. The application can either pre-allocate memory for several elements and repeat the function call in case the function returns AH_WARN_MORE error code or it can call the function without memory pre-allocation, using a NULL pointer. A NULL pointer is a signal for the API that it should provide the number of items without translation of objects.

3. Since AHR and Application both run in one thread, the current model is designed as application-controlled. It means that all operations (including updating of the Horizon cache) are initialised by the application. The **AH_Yield()** function is responsible for updating the Horizon state. Upon a call of an API function, the AHR pulls the data from the receiving thread, applies changes to the Horizon, and performs the requested operation. In that case, no locking is required because AHR and Application access the data sequentially. To avoid the overflow of the CAN buffer (in case the application does not call any function for a longer time), the function **AH_Yield()** should be called by the application regularly if the application does not call other API functions. This will ensure regular processing of incoming messages.
4. The application should be able to start the ADAS Horizon and notify it that it is ready to use its data (register itself in the AHR) or that it does not need information anymore (disconnect from the AHR). These possibilities are realized via Controller's function **AH_ConnectToHorizon()** and **AH_DisconnectFromHorizon().**

5. As it is described above, the content of the Horizon can be updated during several function calls. For example, between requests for link attributes and link attachments. Such situation cannot be prevented, but a special mechanism signalising about changes is introduced. Using *extended identifiers* allows informing the application about changes in content of a link. The Extended identifier consists of two parts: the internal identifier (equal to the one provided by AGMP message) is stored in lower part while the upper part stores some incrementally updated counters. If the entity is replaced (not updated) by a new value, the application will receive the AH_ERR_LINK_DISCARDED error code if it will try to access the old entity by using the old ID. If some items (attributes, profiles or connections) were added, updated or deleted for a link, one of the warning values (AH_WARN_LINK_EDITED, AH_WARN_EDITED_ATT, AH_WARN_EDITED_CONN, AH_WARN_EDITED_PROF) will be returned.

4. Overview of system components

4.1. System initialization (Controller)

The *Controller* is the component containing a set of interfaces used to determine the state of the ADAS horizon at runtime. It is designed to provide control to the ADAS Horizon's working parameters such as level of complexity, data exchange protocol specification, version handling and so on.

All parameters are divided into groups and subgroups, according to the Horizon components, which they belong to or affect. Some parameters are “read only”, but some of them can be changed at runtime (in case the interactive model is supported).

The working parameters of the ADAS Reconstructor system can be divided into three groups:

1. Options *supported* by the Horizon Provider (HP). This set of options depends on the realized version.
2. Options *provided* by the Provider. These options can be considered as a subset of the first group. They can be set by the Horizon Reconstructor (based on the requests from a user application) in case the system works interactively. If no interaction between Provider and Reconstructor is available, the Reconstructor performs filtering of incoming CAN messages.
3. Options *used* by an application. These options can be defined as a subset of the second group.

For example, consider the interactive model, where the Horizon Reconstructor is connected to the Horizon Provider (via the CAN bus) and used by an application (see Fig. 2). At the start of the system, the Reconstructor receives the notification, that the Horizon Provider can provide the information about curvature (curve clusters). This is defined by the options, *supported* by the Horizon.

After the application is connected, the Reconstructor collects the requests, that the application needs information about curve clusters (*Used* options).

The Reconstructor performs a request analysis, recognize, that this request can be handled (Data Store structure allows to store information about curves), and sends to the Provider a notification, that it should transmit the information about curvature, since the application needs it (*Provided* options).

As it was mentioned above, some parameters are “read only”. Some of them can be reset via the configuration file, some depend on the realization only, but they all cannot be reset at runtime. Below, such parameters are called *PROPERTIES*. Changeable parameters are divided into two subgroups: parameters, which have numerical values or defined as structures (they are called *SETTINGS*) and group of parameters, which have Boolean values only (they are called *OPTIONS*).

The behavior of the ADAS Reconstructor system is determined by the AH_MAIN_PROPERTIES. The options and settings are defined for every system’s component. Detailed descriptions of parameters are given below.

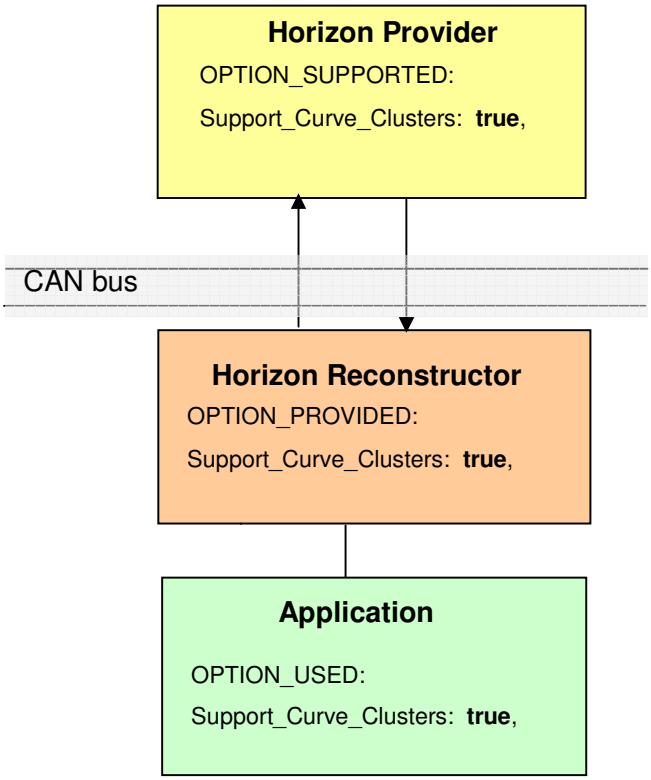

Fig. 2. Three types of options in the system

4.2. Positioner

The *Positioner* is a component designed to provide control to the current position of the vehicle for applications. It contains set of interfaces determining the estimated (real) position and corresponding matched position on the road (optionally the list of such positions sorted by the weight coefficients). The component is also able to do some estimation for the matched position

4.3. Path Reconstructor

The *PathReconstructor* is a component creating the road topology around the vehicle based on the current position and information about links stored in the Horizon cache.

The component gives to a user the possibility to operate with the main or alternative path (if provided) of the Horizon, access the properties of them (profiles), the paths particular elements (links) or associated objects (link attachments).

4.3.1. Link

API *Link* is a basic building block of the Electronic Horizon. It represents (part of) a real-world road. A *Link* is *directed.* Orientation of a *Link* is the same as the direction of the vehicle moving along the link.

Every *Link* has an identifier described by the `AH_LINK_ID` type. In one horizon, the link identifier is a unique value. However, this identifier is realized as a cyclic counter and can and will be reused in next generations of the Electronic Horizon. The API should implement a mechanism that can detect and report out-of-date link identifiers that are used by the application (return code `AH_ERR_LINK_DISCARDED`).

A *Link* has its *geometry*. The *Geometry* of a link is an ordered sequence of Longitude/Latitude pairs (structure `AH_GEOPOINT`) and can be obtained by a `AH_GetLinkPoints()` function call.

Transition from one link to another is described by the structure `AH_LINK_CONNECTION`. This data is available by the function call `AH_GetLinkConnections()`.

Various attributes of the link are stored as *profiles* (structure `AH_PROFILE`) and *attachments* (structure `AH_ATTACHMENT`). Corresponding functions are `AH_GetLinkProfiles()` and `AH_GetLinkAttachment()`. Position of profiles and attachments are given as offsets from the beginning of the link.

Length of the link, direction of traffic flow and route information of the link are available from functions `AH_GetLinkLength()`, `AH_GetLinkOnRoute()` and `AH_GetLinkDriveDirection()`.

Above structures and functions are defined in API Level 1.

4.3.2. Simple Path

Simple Path in API terms is sequence of connected links (without loops).

An ordered sequence of identifiers of links that belong to the path can be obtained with the function call `AH_GetPath()`.

The current position of the vehicle is always on the first link in the path. The *Simple Path* can be considered as one (long and complex) link. All functions that can be used to obtain link data exist in path-variants too.

The *Start of the Path* is the same as the start of the first link on the path. Since the first link on the path is one with a map-matched position, the member `offset` of `AH_POSITION_MATCHED` structure (describing the distance from the start of the matched link) can be viewed as the distance from the start of the path too.

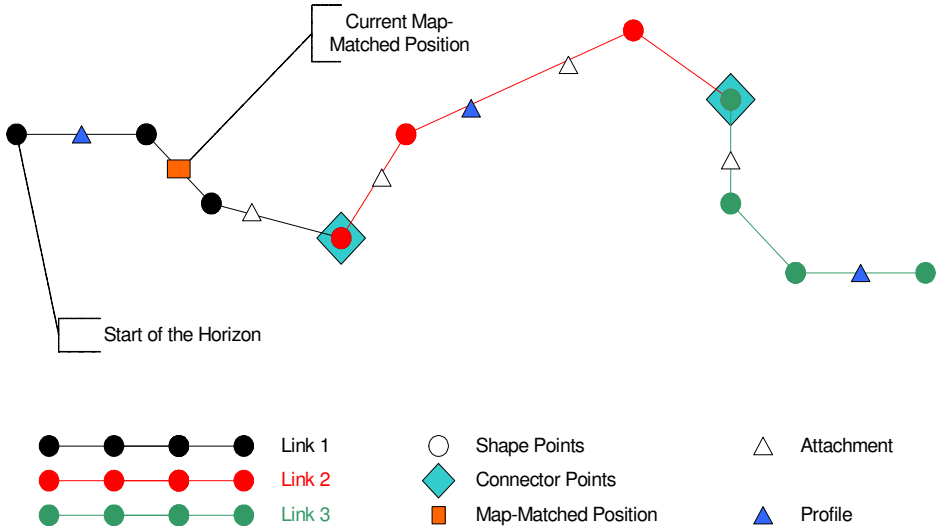

Fig. 3. API Path

Above structures and functions are defined in API Level 2.

4.4. Curve Modeller

The Curve Modeller interface is designed to provide control to the ADAS Horizon's Modeller engine, which is responsible for the creation of a complex curve interpolating.

By the Curve Modeller engine a path (part of a road) in front of the vehicle is modeled by a curve (or series of curves). This curve can be described by a mathematical equation or by the set of mathematical equations allowing defining the curvature parameters like radius or direction for any point on the path. Figure below represents main terms used by the module to describe curvature.

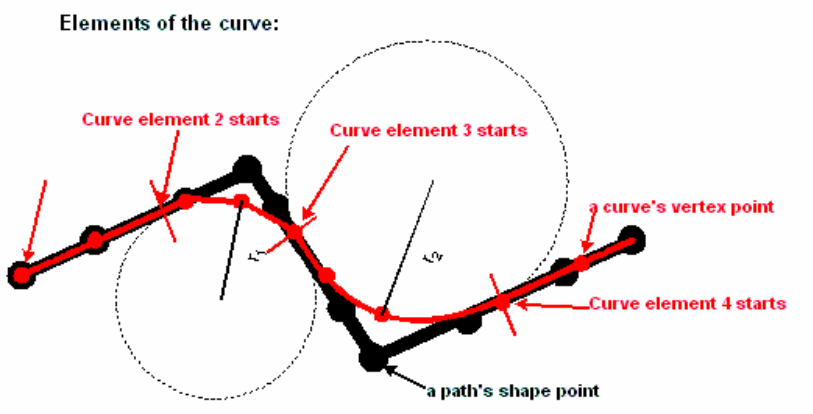

Fig. 4. Main elements of a curve.

5. Summary

This API is considered as the first version of the application programming interface proposed to be used by the ADAS Reconstructor and the ADAS application developer. The current version of the API includes the definitions only for data entities specified as mandatory and defined to be realized within the MAPS&ADAS project. The API is designed in a way that:

1. it allows the ADAS applications fast and safely to receive information from a ADAS Horizon Provider;
2. it allows the applications to access particular data properties as well as complete structures;
3. it specifies all data, which is defined as mandatory for the first version of ADAS Provider systems;
4. it can be easily extended according to new requirements in the future.

Appendix A
Traffic signs supported by ADAS Providers

Code	Icon	Value	Vienna code	Enumerator (API)[1]
0		Traffic light as object	-	LIGHT_LIGHT
A: Danger warning signs				
1		Dangerous bend or bends	A1	WAR_BEND
2		Steep descent	A2	WAR_DESCENT
3		Steep ascent	A3	WAR_ASCENT
4		Carriageway narrows	A4	WAR_NARROW
5		Swing bridge	A5	WAR_BRIDGE
6		Bank	A6	WAR_BANK
7		Uneven road	A7	WAR_UNEVEN
8		Slippery Road	A9	WAR_SLIPPERY
9		Falling rocks	A11	WAR_FALLING_ROCK
10		Pedestrian crossing	A12	WAR_CROSS_PEDESTRIAN
11		Children	A13	WAR_CROSS_CHILDREN
12		Cyclists entering or crossing	A14	WAR_CROSS_BICYCLE

[1] In the enumerator “AH_SIGN_” part of the name is skipped

13		Cattle or other animals crossing	A15	WAR_CROSS_ANIMALS
14		Light signals	A17	WAR_LIGHT
15		Intersection where the priority is prescribed by the general priority rule	A18	WAR_CROSS_SAME_PRIO
16		Intersection with a road of which the user must give way	A19	WAR_CROSS_MY_RIGHTS
17		Roundabout	A22	WAR_ROUNDABOUT
18		Two way traffic resp. oncoming traffic	A23	WAR_TWO_WAY_TRAFFIC
19		Traffic congestion	A24	WAR_CONGESTION
20		Level-crossings with gates	A25	WAR_CROSS_WITH_GATES
21		Unprotected railroad crossing	A26	WAR_CROSS_NO_GATES
22		Airfield	A30	WAR_AIRCRAFT
23		Cross-wing	A31	WAR_WIND
24		Danger	A32	WAR_GENERIC
25		Buses entering or leaving roadway		WAR_CROSS_BUS
26		Risk of ice or snow		WAR_ICE
B: Priority signs				
27		Give way	B1	PRIO_GIVE_WAY
28		Stop	B2	PRIO_STOP
29		Priority road	B3	PRIO_PRIORITY_ROAD
30		End of priority road	B4	PRIO_END_PROIRITY_ROAD
31		Priority for oncoming traffic	B5	PRIO_INCOM_PRIORITY

32		Priority over oncoming traffic	B6	PRIO_PRIORITY_OVER_INCOM
C: Prohibitory or restrictive signs				
33		No entry	C1	REST_NO_ENTRY
34		Closed for all vehicle in both direction	C2	REST_CLOSED
35		Overtaking prohibited	C13a	REST_NO_PASS
36		End of prohibition of overtaking	C17c	REST_NO_PASS_END
37		Overtaking by goods vehicle prohibited	C13b	REST_NO_PASS_TRUCK
38		End of prohibition of overtaking for goods vehicle	C17d	REST_NO_PASS_TRUCK_END
39	N/A	TRUCK/TRUCK overtake prohibited.	N/A	REST_NO_PASS_TRUCK_TRUCK
40	N/A	TRUCK/TRUCK overtake allowed.	N/A	REST_NO_PASS_TRUCK_ TRUCK_END
41		Speed limit sign	C14	REST_SPEED_LIMIT
42		End of speed limit	C17b	REST_SPEED_LIMIT_END
43		End of all local prohibitions	C17a	REST_ALL_END
44		No entry for vehicle having exceeding … tones laden mass	C7	REST_WEIGHT
45		No entry for vehicle having an overall height exceeding ... meters	C6	REST_HEIGHT
46		No entry for vehicle having an overall width exceeding ... meters	C5	REST_WIDTH
47		No entry for vehicle a mass exceeding … tones on one axle	C8	REST_AXLE_LOAD
48		No entry for vehicle or combination of vehicle exceeding … metres in length	C9	REST_LENGTH

49		Driving of vehicle less than … metres apart prohibited	C10	REST_DISTANCE
D: Mandatory				
50		Compulsory roundabout	D3	OBL_ROUNDABOUT
E: Informative				
51		Built-up area entrance sign	E7	INFO_BUILTUP
52		Built-up area exit sign	E8	INFO_BUILTUP_END
53		Maximum speed zone	E9d	INFO_MAX_SPEED_ZONE
54		End of maximum speed zone	E10	INFO_MAX_SPEED_ZONE_END
55		Pedestrian Crossing Prescription	E12	INFO_CROSS_PEDESTRIAN
56		Residential zone	E17a	INFO_RESIDENTIAL_ZONE
57		End of Residential zone	E17b	INFO_RESIDENTIAL_ZONE_END
58		Pedestrian area entrance sign	N/A	INFO_PEDESTRIAN_ZONE
59		Pedestrian area exit sign	N/A	INFO_PEDESTRIAN_ZONE_END
60		Other traffic signs	N/A	OTHER

Appendix B

API design specification
Version 1.0

Table of Contents

API Data Structure Index

1. ADAS Types and definitions

1.1. Basic and Short-named types

Defines

#define MAX_AH_UINT8 255
#define MAX_AH_UINT16 65535
#define MAX_AH_UINT32 4294967295
#define MAX_AH_INT8 127
#define MIN_AH_INT8 -128
#define MIN_AH_INT16 -32768
#define MAX_AH_INT16 32767
#define MIN_AH_INT32 0x80000000
#define MAX_AH_INT32 0x7FFFFFFF
#define AH_TRUE ((AH_BOOL)(1 == 1))
#define AH_FALSE ((AH_BOOL)(1 != 1))

Typedefs

typedef char AH_CHAR
typedef signed char AH_INT8
typedef unsigned char AH_UINT8
typedef signed short AH_INT16
typedef unsigned short AH_UINT16
typedef signed int AH_INT32
typedef unsigned int AH_UINT32

1.1.1. Define Documentation

#define MAX_AH_UINT8 255

The maximum valid value for variables of the AH_UINT8 type

#define MAX_AH_UINT16 65535

The maximum valid value for variables of the AH_UINT16 type

#define MAX_AH_UINT32 4294967295

The maximum valid value for variables of the AH_UINT32 type

#define MAX_AH_INT8 127

The maximum valid value for variables of the AH_INT8 type

#define MIN_AH_INT8 -128

The minimum valid value for variables of the AH_INT8 type

#define MIN_AH_INT16 -32768

The minimum valid value for variables of the AH_INT16 type

#define MAX_AH_INT16 32767

The maximum valid value for variables of the AH_INT16 type

#define MIN_AH_INT32 0x80000000

The minimum valid value for variables of the AH_INT32 type

#define MAX_AH_INT32 0x7FFFFFFF

The maximum valid value for variables of the AH_INT32 type

#define AH_TRUE ((AH_BOOL)(1 == 1))

Defined Boolean value

#define AH_FALSE ((AH_BOOL)(1 != 1))

Defined Boolean value

1.1.2. Typedef Documentation

typedef char AH_CHAR

Size is 1 byte. Possible values range is -128 .. +127

typedef signed char AH_INT8

Size is 1 byte. Possible values range is -128 .. +127

typedef unsigned char AH_UINT8

Size is 1 byte. Possible values range is 0 .. 255

typedef signed short AH_INT16

Size is 2 byte. Possible values range is -32,768 .. +32,767

typedef unsigned short AH_UINT16

Size is 2 byte. Possible values range is 0 .. 65,535

typedef signed int AH_INT32

Size is 4 byte. Possible values range is -2,147,483,648 ..+2,147,483,647

typedef unsigned int AH_UINT32

Size is 4 byte. Possible values range is 0 .. 4,294,967,295

1.2. Content-named types

Defines

#define AH_NOT_SUPPORT 0xFF
#define AH_ALTITUDE_UNDEF 0xFFFF
#define AH_COORDINATE_UNDEF 0xFFFFFFFF
#define AH_DISTANCE_UNDEF 0xFFFFFFFF
#define AH_HEADING_UNDEF 0xFFFF
#define AH_PROBABILITY_UNDEF 0xFFFF
#define AH_SPEED_UNDEF 0xFFFF
#define AH_TIME_UNDEF 0xFFFF
#define AH_SHAPE_UNDEF MIN_AH_INT16
#define AH_VERSION_LENGTH 16
#define AH_COORDINATE_RES 0.00001
#define AH_LONGITUDE_MIN -180
#define AH_LONGITUDE_MAX 180
#define AH_LATITUDE_MIN -90
#define AH_LATITUDE_MAX 90
#define AH_ALTITUDE_RES 5
#define AH_ALTITUDE_MIN -110
#define AH_ALTITUDE_MAX 5000
#define AH_DISTANCE_RES 1
#define AH_HEADING_RES (360.0 / 1020.0)
#define AH_HEADING_MIN 0
#define AH_HEADING_MAX 360
#define AH_PROBABILITY_RES 0.01
#define AH_PROBABILITY_MIN 0
#define AH_PROBABILITY_MAX 100
#define AH_SPEED_RES 0.2
#define AH_SPEED_MIN -12.8
#define AH_SPEED_MAX 89.2
#define AH_TIME_RES 2
#define AH_TIME_MIN 0
#define AH_TIME_MAX 4095

Typedefs

typedef AH_INT16 AH_ALTITUDE
typedef AH_UINT8 AH_BOOL
typedef AH_INT32 AH_COORDINATE
typedef AH_INT32 AH_DISTANCE
typedef AH_INT16 AH_HEADING
typedef AH_UINT8 AH_COUNTER
typedef AH_UINT16 AH_PROBABILITY
typedef AH_INT16 AH_SPEED
typedef AH_INT8 AH_ERROR_CODE
typedef AH_UINT16 AH_TIME
typedef AH_UINT16 AH_LINK_ID
typedef AH_UINT8 AH_PATH_INDEX

1.2.1. Define Documentation

#define AH_NOT_SUPPORT 0xFF

The definition should be used in case Horizon does not support the features described by the enumeration

#define AH_ALTITUDE_UNDEF 0xFFFF

The definition is used to specify undefined value for the AH_ALTITUDE type

#define AH_COORDINATE_UNDEF 0xFFFFFFFF

The definition is used to specify undefined value for the AH_COORDINATE type

#define AH_DISTANCE_UNDEF 0xFFFFFFFF

The definition is used to specify undefined value for the AH_DISTANCE type

#define AH_HEADING_UNDEF 0xFFFF

The definition is used to specify undefined value for the AH_HEADING type

#define AH_PROBABILITY_UNDEF 0xFFFF

The definition is used to specify undefined value for the AH_PROBABILITY type

#define AH_SPEED_UNDEF 0xFFFF

The definition is used to specify undefined value for the AH_SPEED type

#define AH_TIME_UNDEF 0xFFFF

The definition is used to specify undefined value for the AH_TIME type

#define AH_SHAPE_UNDEF MIN_AH_INT16

The definition is used to specify undefined value

#define AH_VERSION_LENGTH 16

Maximal length of the string representing the version of AH Provider

#define AH_COORDINATE_RES 0.00001

Resolution of latitude and longitude (see AH_COORDINATE). Units: degree

#define AH_LONGITUDE_MIN -180

Minimal value of longitude (see AH_COORDINATE)

#define AH_LONGITUDE_MAX 180

Maximal value of longitude (see AH_COORDINATE)

#define AH_LATITUDE_MIN -90

Minimal value of latitude (see AH_COORDINATE)

#define AH_LATITUDE_MAX 90

Maximal value of latitude (see AH_COORDINATE)

#define AH_ALTITUDE_RES 5

Resolution of altitude (see AH_ALTITUDE). Units: meter

#define AH_ALTITUDE_MIN -110

Minimal value of altitude (see AH_COORDINATE)

#define AH_ALTITUDE_MAX 5000

Maximal value of altitude (see AH_COORDINATE)

#define AH_DISTANCE_RES 1

Resolution of distance (see AH_DISTANCE). Units: meter

#define AH_HEADING_RES (360.0 / 1020.0)

Resolution of heading (see AH_HEADING). Units: degree

#define AH_HEADING_MIN 0

Minimal value of heading (see AH_HEADING)

#define AH_HEADING_MAX 360

Maximal value of heading (see AH_HEADING)

#define AH_PROBABILITY_RES 0.01

Resolution of probability (see AH_PROBABILITY). Units: percent's fraction

#define AH_PROBABILITY_MIN 0

Minimal value of probability (see AH_PROBABILITY)

#define AH_PROBABILITY_MAX 100

Maximal value of probability (see AH_PROBABILITY)

#define AH_SPEED_RES 0.2

Resolution of speed (see AH_SPEED). Units: m/s

#define AH_SPEED_MIN -12.8

Minimal value of speed (see AH_SPEED)

#define AH_SPEED_MAX 89.2

Maximal value of speed (see AH_SPEED)

#define AH_TIME_RES 2

Resolution of time (see AH_TIME). Units: ms

#define AH_TIME_MIN 0

Minimal value of time (see AH_TIME)

#define AH_TIME_MAX 4095

Maximal value of time (see AH_TIME)

1.2.2. Typedef Documentation

typedef AH_INT16 AH_ALTITUDE

Height above the Sea level (specified by WGS84). Resolution is AH_ALTITUDE_RES m. AH_ALTITUDE_UNDEF (equal to 0xFFFF) means undefined value.

typedef AH_UINT8 AH_BOOL

A Boolean value. This keyword is an integral type. A variable of this type can have values AH_TRUE and AH_FALSE .

typedef AH_INT32 AH_COORDINATE

Absolute coordinate according to WGS84 (latitude or longitude). Units are AH_COORDINATE_RES degree. Valid values are in the range from -180 degrees west to +180 degrees east (see WGS84) for longitude and from -90 degrees south to +90 degrees north (see WGS84) for latitude.

typedef AH_INT32 AH_DISTANCE

This keyword is an integral type used to measure distances and lengths in the system. The accuracy of measurement depends on the Horizon realization and can be meters or decimeters. To define the current realization, a user should check the correspondent parameter (one of the distance accuracy enumeration value) and divide (if necessary) the received value by this parameter.

Note:

A distance can be also negative e.g. for objects on incoming roads behind

typedef AH_INT16 AH_HEADING

The direction of the vehicle's movement. Measured is fraction of degree (resolution is AH_HEADING_RES degree) in the range from 0 to 360 degree, clockwise, 0 means the direction to the North. In case the value is out of the range, it is undefined.

Note:

The version 1.0 use two-dimension model (horizontal plane) for calculation. In a future it may be enhanced with a vertical component.

typedef AH_UINT8 AH_COUNTER

This type is used in functions, which needs counters or amount of entities.

typedef AH_UINT16 AH_PROBABILITY

Statistical parameter, which specifies, for example, the quality of matching the estimated position to the road network. Unit is percent's fraction equal to AH_PROBABILITY_RES . Possible values are in the range from 0 to 100%.

typedef AH_INT16 AH_SPEED

The rate that the vehicle's position is changing in the direction of the vehicle heading. Unit is AH_SPEED_RES m/s. In case the value is out of the range, it is undefined.

Note:

The version 1.0 use two-dimension model (horizontal plane) for calculation. In a future it may be enhanced with a vertical component.

typedef AH_INT8 AH_ERROR_CODE

Used to specify return value for a function.

typedef AH_UINT16 AH_TIME

The moment in time ("age" of the event, the relative value) or time interval between two events. Units are AH_TIME_RES ms. In case of usage as age: after the maximum value (AH_TIME_MAX ms) is exceeded, the time is set to 0 again.

typedef AH_UINT16 AH_LINK_ID

Identifier of the link. Low-order byte is original link identifier in range 0-127. In high byte, information for tracking data version is stored.

typedef AH_UINT8 AH_PATH_INDEX

Path identifier. Path index 0 denotes Most Probable Path, path index 1 next probable path etc.

Note:

In the version 1.0 only path index 0 is supported by functions.

1.3. Error Handling

#define AH_SUCCESS 0

Used as the return value in case a function is successful.

1.3.1. Warnings

Defines

#define AH_WARN_MORE 1
#define AH_WARN_LINK_EDITED 2
#define AH_WARN_EDITED_ATT 3
#define AH_WARN_EDITED_CONN 4
#define AH_WARN_EDITED_PROF 5

Define Documentation

#define AH_WARN_MORE 1

Function was successful. More data is available.

#define AH_WARN_LINK_EDITED 2

Some of lists of items (connections, attachments, profiles) for required link were edited

#define AH_WARN_EDITED_ATT 3

List of attachments for the link was edited.

Note:

This feature is not supported in version 1.0

#define AH_WARN_EDITED_CONN 4

List of connections for the link was edited.

Note:

This feature is not supported in version 1.0

#define AH_WARN_EDITED_PROF 5

List of profiles for the link was edited.

Note:

This feature is not supported in version 1.0

1.3.2. Errors

Defines

#define AH_ERR_NOT_SUPPORTED -1
#define AH_ERR_INVALID_TYPE -2
#define AH_ERR_NO_RESPONSE -3
#define AH_ERR_UNDER_RANGE -4
#define AH_ERR_OVER_RANGE -5
#define AH_ERR_UNDEFINED -6

```
#define AH_ERR_CHANGED_PARTICULARLY -7
#define AH_ERR_CHANGE_NOT_POSSIBLE -8
#define AH_ERR_INVALID_PARAMETER -9
#define AH_ERR_LINK_DISCARDED -10
#define AH_ERR_NO_MEMORY -11
#define AH_ERR_FAIL -12
```

Define Documentation

#define AH_ERR_NOT_SUPPORTED -1

If requested feature is not supported by the current version of Horizon

#define AH_ERR_INVALID_TYPE -2

If type of method's parameter does not fit the method's specification

#define AH_ERR_NO_RESPONSE -3

If delay of response exceeds the defined possible duration.

#define AH_ERR_UNDER_RANGE -4

If value is valid, but less then the range's minimal threshold.

#define AH_ERR_OVER_RANGE -5

If value is valid, but more then the range's maximal threshold.

#define AH_ERR_UNDEFINED -6

If required information does not exist.

#define AH_ERR_CHANGED_PARTICULARLY -7

Some properties of component was changed.

#define AH_ERR_CHANGE_NOT_POSSIBLE -8

Changing of the component options is not possible in runtime.

#define AH_ERR_INVALID_PARAMETER -9

If input parameter is invalid or NULL

#define AH_ERR_LINK_DISCARDED -10

Requested link was discarded in horizon or it does not exists. New link with the same internal id has taken its place.

#define AH_ERR_NO_MEMORY -11

Not enough memory to complete the operation.

#define AH_ERR_FAIL -12

Function was not executed successfully for some other reasons.

1.4. Defines used in function declarations

#define IN
#define OUT
#define INOUT
#define PRIVATE static
#define PUBLIC

Define Documentation

#define IN

To be used as specifier for function arguments - input parameter.

#define OUT

To be used as specifier for function arguments - output parameter.

#define INOUT

To be used as specifier for function arguments - input/output parameter.

#define PRIVATE static

To be used as local function specifier.

#define PUBLIC

To be used as exported function specifier.

2. Components

2.1. Controller

2.1.1. Controller enumerations

enum AH_DRIVING_SIDE
enum AH_TIME_ZONE
enum AH_ACCURACY
enum AH_MODEL_CURVE
enum AH_MODEL_PATH
enum AH_LENGTH_UNITS
enum AH_SP_DELTA_UNITS
enum AH_PROVIDERS_MAP
enum AH_PROVIDERS_HORIZON
enum AH_PARAMETERS_TYPE

Enumeration Type Documentation

enum AH_DRIVING_SIDE

The enumeration specifies kinds of driving sides

Enumerator:

AH_DRIVING_SIDE_LEFT The default driving side in the country is left

AH_DRIVING_SIDE_RIGHT The default driving side in the country is right

AH_DRIVING_SIDE_UNDEF Driving side is undefined

enum AH_TIME_ZONE

The enumeration determines the difference in hours between the time on the host computer and Universal Coordinated Time (UTC).

Enumerator:

AH_GMT_MINUS_12 (UTC - 12:00): Eniwetok, Kwajalein

AH_GMT_MINUS_11 (UTC - 11:00): Midway Island, Samoa

AH_GMT_MINUS_10 (UTC - 10:00): Hawaii

AH_GMT_MINUS_9 (UTC - 9:00): Alaska

AH_GMT_MINUS_8 (UTC - 8:00): Pacific time

AH_GMT_MINUS_7 (UTC - 7:00): Mountain time, Arizona

AH_GMT_MINUS_6 (UTC - 6:00): Central time, Saskatchewan, Mexico City, Tegucigalpa

AH_GMT_MINUS_5 (UTC - 5:00): Eastern time , Indiana, Bogota, Lima, Quito

AH_GMT_MINUS_4 (UTC - 4:00): Atlantic time, Caracas, La Paz

AH_GMT_MINUS_3 (UTC - 3:00): Brasilia, Buenos Aires, Georgetown

AH_GMT_MINUS_2 (UTC - 2:00): Mid Atlantic

AH_GMT_MINUS_1 (UTC - 1:00): Azores, Cape Verde Islands

AH_GMT Greenwich Mean Time (GMT), Coordinated Universal Time (UTC). UTC + 0:00: Dublin, Edinburgh, Lisbon, London, Casablanca, Monrovia

AH_GMT_PLUS_1 (UTC + 1:00): Brussels, Copenhagen, Madrid, Paris, Vilnius, Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna, Belgrade, Pozsony, Budapest, Ljubljana, Prague

AH_GMT_PLUS_2 (UTC + 2:00): Eastern Europe, Athens, Instanbul, Minsk, Israel, Cairo, Harare, Pretoria

AH_GMT_PLUS_3 (UTC + 3:00): Baghdad, Kuwait, Riyadh, Moscow, St. Petersburg, Volgograd

AH_GMT_PLUS_4 (UTC + 4:00): Abu Dhabi, Muscat

AH_GMT_PLUS_5 (UTC + 5:00): Islamabad, Karachi, Tashkent

AH_GMT_PLUS_6 (UTC + 6:00): Akmola, Almaty, Dhaka

AH_GMT_PLUS_7 (UTC + 7:00): Bangkok, Hanoi, Jakarta

AH_GMT_PLUS_8 (UTC + 8:00): Hong Kong SAR, Beijing, Chongqing, Urumqi

AH_GMT_PLUS_9 (UTC + 9:00): Osaka, Sapporo, Tokyo

AH_GMT_PLUS_10 (UTC + 10:00): Brisbane, Hobart, Guam, Port Moresby

AH_GMT_PLUS_11 (UTC + 11:00): Magadan, Solomon Islands, New Caledonia

AH_GMT_PLUS_12 (UTC + 12:00): Auckland, Wellington, Fiji Islands, Kamchatka, Marshall Islands

AH_TIME_ZONE_UNDEF Invalid (undefined) time zone

enum AH_ACCURACY

The enumeration specifies accuracy categories. Detailed description of accuracy classes can be found in the "Interface and Data Entity Specifications. Part 1 - General Specifications" document, Chapter 4.2.2.

Enumerator:

AH_ACCURACY_CLASS_1 Error is in a range from 0 till 2^k units, highest accuracy, lowest error class

AH_ACCURACY_CLASS_2 Error is in a range from 2^k till 2^(k+1) units

AH_ACCURACY_CLASS_3 Error is in a range from 2^(k+1) till 2^(k+2) units

AH_ACCURACY_CLASS_4 Error is in a range from 2^(k+3) till 2^(k+3) units, lowest accuracy

AH_ACCURACY_UNDEF Accuracy is undefined/unknown, leading to worst case assumption of highest error class

enum AH_MODEL_CURVE

The enumeration specifies types of the model of Curve Modeller.

Enumerator:

AH_CURVE_NOT_SUPPORTED Online curvature calculation is not supported by Provider

AH_HEADING_CHANGE Heading change is provided

AH_LINEAR Provider supports Simple linear interpolation

AH_B_SPLINE Provider supports B-spline interpolation

AH_BEZIER Provider supports Bezier interpolation

AH_CLOTHOID Provider supports interpolation by clothoids (only)

AH_CLOTHOID_COMPLEX Provider supports interpolation by clothoids, lines, circles

AH_MODEL_CURVE_OTHER Provider supports another (not mentioned above) interpolation model

AH_MODEL_CURVE_MODEL_UNDEF The model of Curve Modeller is undefined

enum AH_MODEL_PATH

The enumeration specifies possible Horizon models.

Enumerator:

AH_POINTS Point Horizon model is provided or supported

AH_SINGLE_PATH Single (Most Probable) path Horizon model is provided or supported

AH_SINGLE_PATH_WITH_STUBS Single (Most Probable) path with stubs (information about outgoing roads on crossings) model of Horizon is provided or supported

AH_MULTIPATHS Multipath Horizon model is provided or supported. It can be complete Horizon tree or defined (more than one) number of alternative paths within the Horizon tree

AH_MODEL_PATH_UNDEF The model of Path Reconstructor is undefined

enum AH_LENGTH_UNITS

One of these values is used to specify the current resolution for lengths and distances measurement

Enumerator:

AH_METERS Resolution is 1 meter. Fractions are not used; truncation from float to the lowest integer value is performed.

AH_DECIMETERS Resolution is 0.1 meter. Values of the AH_DISTANCE type are integer; in order to receive actual value a user should divide them to 0.1.

AH_LENGTH_UNITS_UNDEF Current resolution is undefined.

enum AH_SP_DELTA_UNITS

One of these values is used to specify units for transmitting of delta values for shape points on CAN protocol level.

Enumerator:

AH_SP_DELTA_METERS Meters are used.

AH_SP_DELTA_DEGREE Degree fractions are used.

AH_SP_DELTA_UNDEF information about used units is not available.

enum AH_PROVIDERS_MAP

This enumeration specifies possible providers of maps

Enumerator:

AH_NAVTEQ Provider of map is NAVTEQ

AH_TELEATLAS Provider of map is Tele Atlas

AH_ZENRIN Provider of map is ZENRIN

AH_PROVIDERS_MAP_OTHER Other provider of map

AH_PROVIDERS_MAP_UNDEF Provider of map is undefined

enum AH_PROVIDERS_HORIZON

This enumeration specifies possible providers of horizons

Enumerator:

AH_PROVIDER_BLAUPUNKT Provider of horizon is Blaupunkt

AH_PROVIDER_NAVIGON Provider of horizon is NAVIGON

AH_PROVIDER_NAVTEQ Provider of horizon is NAVTEQ

AH_PROVIDER_SIEMENS Provider of horizon is Siemens

AH_PROVIDERS_HORIZON_OTHER Other provider of horizon

AH_PROVIDERS_HORIZON_UNDEF Provider of horizon is undefined

enum AH_PARAMETERS_TYPE

Type of parameters used by the Controller component

Enumerator:

AH_SUPPORTED Parameters supported by Reconstructor

AH_PROVIDED Parameters provided to Reconstructor by ADAS Provider

AH_PARAMETERS_TYPE_UNDEF Invalid / undefined type of parameters

2.1.2. Controller structures

struct AH_SETTINGS_POSITIONER
struct AH_SETTINGS_RECONSTRUCTOR
struct AH_SETTINGS_MODELLER
struct AH_PROPERTIES_MAIN
struct AH_METADATA
struct AH_METADATA_EXT

2.1.3. Controller functions

AH_ERROR_CODE AH_ConnectToHorizon (void)
AH_ERROR_CODE AH_DisconnectFromHorizon (void)
AH_ERROR_CODE AH_GetMetadata
(OUT const AH_METADATA **metadata)
AH_ERROR_CODE AH_GetMetadataExt
(OUT const AH_METADATA_EXT **metadata_ext)
AH_ERROR_CODE AH_GetPropertiesMain
(IN AH_PARAMETERS_TYPE type,
OUT const AH_PROPERTIES_MAIN **main_properties)
AH_ERROR_CODE AH_SetPropertiesMain
(IN AH_PROPERTIES_MAIN *main_properties)
AH_ERROR_CODE AH_GetSettingsModeller
(IN AH_PARAMETERS_TYPE type,
OUT const AH_SETTINGS_MODELLER **cm_settings)
AH_ERROR_CODE AH_SetSettingsModeller
(IN AH_SETTINGS_MODELLER *cm_settings)
AH_ERROR_CODE AH_GetSettingsPositioner
(IN AH_PARAMETERS_TYPE type,
OUT const AH_SETTINGS_POSITIONER **pos_settings)
AH_ERROR_CODE AH_SetSettingsPositioner
(IN AH_SETTINGS_POSITIONER *pos_settings)
AH_ERROR_CODE AH_GetSettingsReconstructor
(IN AH_PARAMETERS_TYPE type,
OUT const AH_SETTINGS_RECONSTRUCTOR **pr_settings)
AH_ERROR_CODE AH_SetSettingsReconstructor
(IN AH_SETTINGS_RECONSTRUCTOR *pr_settings)
AH_ERROR_CODE AH_Yield (void)

Function Documentation

AH_ERROR_CODE AH_ConnectToHorizon (void)

Connect application to the ADAS Horizon Reconstructor.

Returns:

AH_SUCCESS in case the connection is successful, otherwise returns one of the error codes.

AH_ERROR_CODE AH_DisconnectFromHorizon (void)

Disconnect application from the ADAS Horizon Reconstructor.

Returns:

AH_SUCCESS in case the disconnection is successful, otherwise returns one of the error codes.

AH_ERROR_CODE AH_GetMetadata
(OUT const AH_METADATA ** *metadata*)

Gets main specification of the system.

Parameters:

metadata Pointer to pointer to the AH_METADATA structure that contains the description of the system.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case the current version does not store metadata

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_GetMetadataExt
(OUT const AH_METADATA_EXT ** *metadata_ext*)

Gets extended (full) description of the system in form of metadata.

Parameters:

metadata_ext Pointer to pointer to the AH_METADATA_EXT structure that contains the metadata.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case the current version does not store extended metadata

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_GetPropertiesMain
(IN AH_PARAMETERS_TYPE *type*,
OUT const AH_PROPERTIES_MAIN ** *main_properties*)

Gets information about supported or provided properties of the system.

Parameters:

type Type of the settings which are requested. One of the AH_PARAMETERS_TYPE enumeration value.

main_properties Pointer to pointer to the AH_PROPERTIES_MAIN that contains the main properties of the system (common length, maximal distance, interval for reference points and so on)

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case the current version does not make differentiation between supported and provided settings.

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_SetPropertiesMain
(IN AH_PROPERTIES_MAIN * *main_properties*)

Attempt to set new main settings for the Horizon.

Parameters:

main_properties Pointer to the AH_PROPERTIES_MAIN structure that contains new desired properties of Horizon.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case it is not possible to change properties in runtime

AH_ERR_CHANGED_PARTICULARLY in case only some of properties was changed by the procedure

AH_ERR_CHANGE_NOT_POSSIBLE in case the new properties can not be assigned to the system since some components needs old properties.

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_GetSettingsModeller
(IN AH_PARAMETERS_TYPE *type*,
OUT const AH_SETTINGS_MODELLER ** *cm_settings*)

Gets information about supported or provided settings of the Curve Modeller component.

Parameters:

type Type of the settings which are requested. One of the AH_PARAMETERS_TYPE enumeration value.

cm_settings Pointer to pointer to the AH_SETTINGS_MODELLER that contains one of the desired settings.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case the current version does not make differentiation between supported and provided settings.

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_SetSettingsModeller
(IN AH_SETTINGS_MODELLER * *cm_settings*)

Attempt to set new settings to the Curve Modeller.

Parameters:

cm_settings Pointer to the AH_SETTINGS_MODELLER structure that contains new desired settings for Curve Modeller.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case it is not possible to change properties in runtime

AH_ERR_CHANGED_PARTICULARLY in case only some of properties was changed by the procedure

AH_ERR_CHANGE_NOT_POSSIBLE in case the new properties can not be assigned to the Curve Modeller since other components needs old properties.

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_GetSettingsPositioner
(IN AH_PARAMETERS_TYPE *type*,
OUT const AH_SETTINGS_POSITIONER ** *pos_settings*)

Gets information about supported or provided settings of the Positioner component.

Parameters:

type Type of the settings, which are requested. One of the AH_PARAMETERS_TYPE enumeration value.

pos_settings Pointer to pointer to the AH_SETTINGS_POSITIONER that contains one of the desired settings.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case the current version does not make differentiation between supported and provided settings.

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_SetSettingsPositioner (IN AH_SETTINGS_POSITIONER * *pos_settings*)

Attempt to change provided setting of the Positioner component.

Parameters:

pos_settings Pointer to the AH_SETTINGS_POSITIONER structure that contains new desired settings for the Positioner.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case it is not possible to change properties in runtime

AH_ERR_CHANGED_PARTICULARLY in case only some of properties was changed by the procedure

AH_ERR_CHANGE_NOT_POSSIBLE in case the new properties can not be assigned to the Positioner since other components needs old properties.

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_GetSettingsReconstructor (IN AH_PARAMETERS_TYPE *type*, OUT const AH_SETTINGS_RECONSTRUCTOR ** *pr_settings*)

Gets information about supported or provided settings of the Path Reconstructor.

Parameters:

type Type of the settings which are requested. One of the AH_PARAMETERS_TYPE enumeration value.

pr_settings Pointer to pointer to the AH_SETTINGS_RECONSTRUCTOR that contains one of the desired settings.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case the current version does not make differentiation between supported and provided settings.

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_SetSettingsReconstructor
(IN AH_SETTINGS_RECONSTRUCTOR * *pr_settings*)

Attempt to change provided setting of the Path Reconstructor component.

Parameters:

pr_settings Pointer to the AH_SETTINGS_RECONSTRUCTOR structure that contains new desired settings for the Path Reconstructor.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case it is not possible to change properties in runtime

AH_ERR_CHANGED_PARTICULARLY in case only some of properties was changed by the procedure

AH_ERR_CHANGE_NOT_POSSIBLE in case the new properties can not be assigned to the Reconstructor since other components needs old properties.

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_Yield (void)

Allows to Horizon to update the cache state

Returns:

If successful, the routine returns AH_SUCCESS. On failure, returns one of the error values.

2.2. Path Reconstructor

2.2.1. Path Reconstructor enumerations

enum AH_DRIVE_DIRECTION
enum AH_JUNCTION_TYPE
enum AH_SIGN_LOCATION
enum AH_SIGN_TYPE
enum AH_SIGN_VALUE
enum AH_ATTACHMENT_TYPE
enum AH_PROFILE_TYPE
enum AH_ROAD_CLASS
enum AH_WAY_FORM
enum AH_TRANSITION
enum AH_GIVEWAY
enum anonymous

Enumeration Type Documentation

enum AH_DRIVE_DIRECTION

Enumeration specifies the direction in which the driving is allowed

Enumerator:

AH_DR_DIR_OPPOSITE Driving is not allowed in this direction

AH_DR_DIR_BOTH_WAYS Driving is allowed in both directions

AH_DR_DIR_ONE_WAY Driving is not allowed in opposite direction

AH_DR_DIR_NOT_ALLOWED Driving is not allowed in any direction

AH_DR_DIR_UNDEF direction of driving is undefined

enum AH_JUNCTION_TYPE

Type of the Junction.

Enumerator:

AH_MINI_ROUNDABOUT Roundabout

AH_RAILWAY_CROSSING Crossing with railway

AH_BORDER_CROSSING Country boarder is crossed

AH_JUNCTION_TYPE_UNDEF Undefined / invalid type

enum AH_SIGN_LOCATION

Location of the Traffic Sign.

Enumerator:

AH_SIGN_LOCATION_LEFT The traffic sign is located on left side of the road

AH_SIGN_LOCATION_RIGHT The traffic sign is located on right side of the road

AH_SIGN_LEFT_AND_RIGHT The traffic sign is located on both sides of the road

AH_SIGN_LOCATION_OVERHEAD The traffic sign is located over the road

AH_SIGN_LOCATION_OTHER Other location of the traffic sign

AH_SIGN_LOCATION_UNDEF Location is not defined

enum AH_SIGN_TYPE

Type of the traffic sign (according to "CONVENTION ON ROAD SIGNS AND SIGNALS", Vienna, 08.11.1968)

Enumerator:

AH_SIGN_TYPE_WAR_DANGER Danger warning signs intended to warn road-users of a danger on the road and to inform them of its nature

AH_SIGN_TYPE_REG_PRIORITY Priority signs notifying about the special rules of priority at intersections

AH_SIGN_TYPE_REG_PROHIBITORY Prohibitory or restrictive signs

AH_SIGN_TYPE_REG_MANDATORY Mandatory signs

AH_SIGN_TYPE_REG_SPECIAL Special regulation signs

AH_SIGN_TYPE_INFO_SERVICE Information, facilities or service signs

AH_SIGN_TYPE_INFO_DIR_ADV Advance direction signs

AH_SIGN_TYPE_INFO_DIR Direction signs

AH_SIGN_TYPE_INFO_IDENT_ROAD Road identification signs

AH_SIGN_TYPE_INFO_IDENT_PLACE Place identification signs

AH_SIGN_TYPE_INFO_CONFIRM Confirmatory signs

AH_SIGN_TYPE_ADDITIONAL Additional panels

AH_SIGN_TYPE_OTHER Other type of signs, not declared by Vienna Convention

AH_SIGN_TYPE_UNDEF Invalid / undefined type of sign

enum AH_SIGN_VALUE

Traffic Sign.

Enumerator:

AH_SIGN_LIGNT_LIGHT Traffic light as light

AH_SIGN_WAR_BEND Dangerous bend or bends

AH_SIGN_WAR_DESCENT Steep descent

AH_SIGN_WAR_ASCENT Steep ascent

AH_SIGN_WAR_NARROW Carriageway narrows

AH_SIGN_WAR_BRIDGE Swing bridge

AH_SIGN_WAR_BANK Bank

AH_SIGN_WAR_UNEVEN Uneven road

AH_SIGN_WAR_SLIPPERY Slippery Road

AH_SIGN_WAR_FALLING_ROCK Falling rocks

AH_SIGN_WAR_CROSS_PEDESTRIAN Pedestrian crossing

AH_SIGN_WAR_CROSS_CHILDREN Children

AH_SIGN_WAR_CROSS_BICYCLE Cyclists entering or crossing

AH_SIGN_WAR_CROSS_ANIMALS Cattle or other animals crossing

AH_SIGN_WAR_LIGHT Light signals

AH_SIGN_WAR_CROSS_SAME_PRIO Intersection where the priority is prescript by the general priority rule

AH_SIGN_WAR_CROSS_MY_RIGHTS Intersection with a road the user of which must give way

AH_SIGN_WAR_ROUNDABOUT Roundabout

AH_SIGN_WAR_TWO_WAY_TRAFFIC Two way traffic resp. oncoming traffic

AH_SIGN_WAR_CONGESTION Traffic congestion

AH_SIGN_WAR_CROSS_WITH_GATES Level-crossings with gates

AH_SIGN_WAR_CROSS_NO_GATES Unprotected railroad crossing

AH_SIGN_WAR_AIRCRAFT Airfield

AH_SIGN_WAR_WIND Cross-wing

AH_SIGN_WAR_GENERIC Danger

AH_SIGN_WAR_CROSS_BUS Buses entering or leaving roadway

AH_SIGN_WAR_ICE Risk of ice or snow

AH_SIGN_PRIO_GIVE_WAY Give way

AH_SIGN_PRIO_STOP Stop

AH_SIGN_PRIO_PRIORITY_ROAD Priority road

AH_SIGN_PRIO_END_PROIRITY_ROAD End of priority road

AH_SIGN_PRIO_INCOM_PRIORITY Priority for oncoming traffic

AH_SIGN_PRIO_PRIORITY_OVER_INCOM Priority over oncoming traffic

AH_SIGN_REST_NO_ENTRY No entry

AH_SIGN_REST_CLOSED Closed for all vehicle in both direction

AH_SIGN_REST_NO_PASS Overtaking prohibited

AH_SIGN_REST_NO_PASS_END End of prohibition of overtaking

AH_SIGN_REST_NO_PASS_TRACK Overtaking by goods vehicle prohibited

AH_SIGN_REST_NO_PASS_TRACK_END End of prohibition of overtaking for goods vehicle

AH_SIGN_REST_NO_PASS_TRACK_TRACK TRUCK/TRUCK overtake prohibited

AH_SIGN_REST_NO_PASS_TRACK_TRACK_END TRUCK/TRUCK overtake allowed

AH_SIGN_REST_SPEED_LIMIT Speed limit sign

AH_SIGN_REST_SPEED_LIMIT_END End of speed limit

AH_SIGN_REST_ALL_END End of all local prohibitions

AH_SIGN_REST_WEIGHT No entry for vehicle having exceeding … tones laden mass

AH_SIGN_REST_HEIGHT No entry for vehicle having an overall height exceeding ... meters

AH_SIGN_REST_WIDTH No entry for vehicle having an overall width exceeding ... meters

AH_SIGN_REST_AXLE_LOAD No entry for vehicle a mass exceeding … tones on one axle

AH_SIGN_REST_LENGTH No entry for vehicle or combination of vehicle exceeding … meters in length

AH_SIGN_REST_DISTANCE Driving of vehicle less than … meters apart prohibited

AH_SIGN_OBL_ROUNDABOUT Compulsory roundabout

AH_SIGN_INFO_BUILTUP Built-up area entrance sign

AH_SIGN_INFO_BUILTUP_END Built-up area exit sign

AH_SIGN_INFO_MAX_SPEED_ZONE Maximum speed zone

AH_SIGN_INFO_MAX_SPEED_ZONE_END End of maximum speed zone

AH_SIGN_INFO_CROSS_PEDESTRIAN Pedestrian crossing prescription

AH_SIGN_INFO_RESIDENTIAL_ZONE Residential zone

AH_SIGN_INFO_RESIDENTIAL_ZONE_END End of residential zone

AH_SIGN_INFO_PEDESTRIAN_ZONE Pedestrian area entrance sign

AH_SIGN_INFO_PEDESTRIAN_ZONE_END Pedestrian area exit sign

AH_SIGN_OTHER Other traffic signs

AH_SIGN_UNDEF Invalid / undefined sign

enum AH_ATTACHMENT_TYPE

Type of the Link Attachment.

Enumerator:

AH_LA_HEADING_CHANGE Heading change

AH_LA_TRAFFIC_SIGN Traffic sign. See also AH_SIGN

AH_LA_TYPE_CNT Counter to define max number of the enumerator

AH_LA_TYPE_UNDEF Invalid / undefined type of attachment

enum AH_PROFILE_TYPE

Type of the Profile.

Enumerator:

AH_PROFILE_FREEWAY Free way

AH_PROFILE_WAY_FORM Form of way

AH_PROFILE_ROAD_CLASS Road class. This is the value of bits 4-0 of LINK_PARAM field.

AH_PROFILE_SPEED_LIMIT Speed limit

AH_PROFILE_CURVATURE Curvature

AH_PROFILE_LANE_NUMBER Number of lanes. This is the value if bits 7-5 of LINK_PARAM field.

AH_PROFILE_LANE_NUMBER_OPPOSITE Number of lanes in opposite direction

AH_PROFILE_IN_BUILD_AREA Build-in area defined by domestic legislation

AH_PROFILE_FERRY_CONNECTION Ferry connections

AH_PROFILE_SLOPE Slope

AH_PROFILE_TYPE_CNT Counter to define max number of the enumerator

AH_PROFILE_UNDEF Invalid / undefined type of profile

enum AH_ROAD_CLASS

Class of the road. This is the value of bits 4-0 of LINK_PARAM field.

Note:

This classification is different for different Map and Horizon providers, therefore "no-named" classes are used. Application should take care how to process the information dependent on the type of Horizon provider.

Enumerator:

AH_ROAD_CLASS_1 Most important roads are included

AH_ROAD_CLASS_2 Self-explanatory

AH_ROAD_CLASS_3 Self-explanatory

AH_ROAD_CLASS_4 Self-explanatory

AH_ROAD_CLASS_5 Self-explanatory

AH_ROAD_CLASS_6 Self-explanatory

AH_ROAD_CLASS_7 Self-explanatory

AH_ROAD_CLASS_8 Lowest road categories are included

AH_ROAD_CLASS_OTHER Specific and non-public roads

AH_ROAD_CLASS_UNDEF Invalid / undefined road class

enum AH_WAY_FORM

Type of the way.

Enumerator:

AH_WAY_FORM_MOTORWAY A road specially designed and built for motor traffic, which does not serve properties bordering on it

AH_WAY_FORM_MULTIPLE_CARRIAGE A road comprises several carriageways clearly separated from one another by, for example, a dividing strip

AH_WAY_FORM_SINGLE_CARRIAGE A road normally used by vehicular traffic without clearly separation bitween traffic direction

AH_WAY_FORM_ROUNDABOUT A part of road is roundabout

AH_WAY_FORM_SLIP_ROAD Slip road

AH_WAY_FORM_UNDEF Invalid / undefined form of way

enum AH_TRANSITION

Description of the transition between links.

Enumerator:

AH_TRANSITION_NOT_POSSIBLE Driving is physically not possible

AH_TRANSITION_ALLOWED Driving is possible and allowed

AH_TRANSITION_NOT_ALLOWED Driving is possible but not allowed

AH_TRANSITION_UNDEF Invalid / undefined transition

enum AH_GIVEWAY

Right of way.

Enumerator:

AH_GIVEWAY_NONE No special rights are defined at intersection.

AH_GIVEWAY_RIGHT_OF_WAY drivers of vehicles moving along or coming from such other roads are required to give way to vehicles moving along that road

AH_GIVEWAY_GIVE_WAY drivers must give way to vehicles on the road they are approaching

AH_GIVEWAY_UNDEF Undefined rights

anonymous enum

Undefined values.

Enumerator:

AH_UNDEF_LINK_LANES Undefined number of lanes

AH_UNDEF_LINK_TYPE Undefined type of link

2.2.2. Path Reconstructor structures

struct AH_DESCRIPTION
struct AH_SIGN
struct AH_ATTACHMENT
struct AH_PROFILE
struct AH_LINK_CONNECTION

2.2.3. Path Reconstructor functions

```
AH_ERROR_CODE AH_GetPath
                    (AH_PATH_INDEX pathIndex,
                     AH_COUNTER *size,
                     AH_LINK_ID *linkId)
AH_ERROR_CODE AH_GetLinkLength
                    (IN    AH_LINK_ID linkId,
                     OUT AH_DISTANCE *length)
AH_ERROR_CODE AH_GetLinkOnRoute
                    (IN    AH_LINK_ID linkId,
                     OUT AH_BOOL *onRoute)
AH_ERROR_CODE AH_GetLinkDriveDirection
                     (IN    AH_LINK_ID linkId,
                      OUT AH_DRIVE_DIRECTION *driveDir)
AH_ERROR_CODE AH_GetLinkAttachments
                     (IN AH_LINK_ID linkId,
                      IN AH_ATTACHMENT_TYPE type,
                      INOUT AH_COUNTER *size,
                      OUT AH_ATTACHMENT *att)
AH_ERROR_CODE AH_GetLinkProfiles
                     (IN AH_LINK_ID linkId,
                      IN AH_PROFILE_TYPE type,
                      INOUT AH_COUNTER *size,
                         OUT AH_PROFILE *prof)
AH_ERROR_CODE AH_GetLinkConnections
                     (IN AH_LINK_ID linkId,
                      INOUT AH_COUNTER *size,
                        OUT AH_LINK_CONNECTION *conn)
AH_ERROR_CODE AH_GetLinkConnection
                     (IN AH_LINK_ID linkId,
                      OUT AH_LINK_CONNECTION *conn)
```

AH_ERROR_CODE AH_GetLinkPoints
(IN AH_LINK_ID linkId,
INOUT AH_COUNTER *size,
INOUT AH_GEOPOINT *pt)
AH_ERROR_CODE AH_GetPathLength
(IN AH_PATH_INDEX pathIndex,
OUT AH_DISTANCE *length)
AH_ERROR_CODE AH_GetPathOnRoute
(IN AH_PATH_INDEX pathIndex,
OUT AH_BOOL *onRoute)
AH_ERROR_CODE AH_GetPathAttachments
(IN AH_PATH_INDEX pathIndex,
IN AH_ATTACHMENT_TYPE type,
INOUT AH_COUNTER *size,
OUT AH_ATTACHMENT *att)
AH_ERROR_CODE AH_GetPathProfiles
(IN AH_PATH_INDEX pathIndex,
IN AH_PROFILE_TYPE type,
INOUT AH_COUNTER *size,
OUT AH_PROFILE *prof)
AH_ERROR_CODE AH_GetPathConnections
(IN AH_PATH_INDEX pathIndex,
INOUT AH_COUNTER *size,
OUT AH_LINK_CONNECTION *conn,
OUT AH_DISTANCE *dist)
AH_ERROR_CODE AH_GetPathPoints
(IN AH_PATH_INDEX pathIndex,
INOUT AH_COUNTER *size,
OUT AH_GEOPOINT *pt)

Function Documentation

AH_ERROR_CODE AH_GetPath
(AH_PATH_INDEX *pathIndex*,
AH_COUNTER * *size*,
AH_LINK_ID * *linkId*)

Retrieves ordered list of links that form one path. Currently, only path_index = 0 (most probable path) is supported.

Parameters:

pathIndex Index of path whose links are to be retrieved.

size Pointer to array size. Initially, the variable must contain capacity of the output array. On output, it will contain the information, how many members of output array were filled.

linkId Pointer to array of link identifiers that will be filled with identifiers of path links. If this pointer is NULL, *size will be set to size of array necessary to hold all available data.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_MORE Function was successful, but there are more items available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERROR_CODE AH_GetLinkLength
(IN AH_LINK_ID *linkId*,
OUT AH_DISTANCE * *length*)

Retrieves length of the link.

Parameters:

linkId Identifier of the link.

length Pointer to variable that will receive length of the link. Length will be expressed in current distance units.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_LINK_EDITED Function was successful; new link data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERR_LINK_DISCARDED Link is no more available.

AH_ERROR_CODE AH_GetLinkOnRoute
(IN AH_LINK_ID *linkId*,
OUT AH_BOOL * *onRoute*)

Checks if link is part of the calculated route.

Parameters:

linkId Identifier of the link.

onRoute Pointer to variable that will receive on-route flag.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_LINK_EDITED Function was successful; new link data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERR_LINK_DISCARDED Link is no more available.

AH_ERROR_CODE AH_GetLinkDriveDirection
(IN AH_LINK_ID *linkId*,
OUT AH_DRIVE_DIRECTION * *driveDir*)

Gets information of direction of traffic flow on a link.

Parameters:

linkId Identifier of the link.

driveDir Pointer to variable that will receive link traffic flow value.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_LINK_EDITED Function was successful; new link data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERR_LINK_DISCARDED Link is no more available.

AH_ERROR_CODE AH_GetLinkAttachments
(IN AH_LINK_ID *linkId*,
IN AH_ATTACHMENT_TYPE *type*,
INOUT AH_COUNTER * *size*,
OUT AH_ATTACHMENT * *att*)

Retrieves attachments of the link.

Parameters:

linkId Identifier of the link.

type type of attachment to be retrieved. If AH_LA_TYPE_UNDEF type is used, all attachments will be taken.

size Pointer to array size. Initially, the variable must contain capacity of the output array. On output, it will contain how many members of output array were filled.

att Pointer to array to be filled with link attachments. If this pointer is NULL, *size will be set to size of array necessary to hold all the available data.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_MORE Function was successful; more data is available.

AH_WARN_LINK_EDITED Function was successful; new link data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERR_LINK_DISCARDED Link is no more available.

AH_ERROR_CODE AH_GetLinkProfiles
(IN AH_LINK_ID *linkId*,
IN AH_PROFILE_TYPE *type*,
INOUT AH_COUNTER * *size*,
OUT AH_PROFILE * *prof*)

Retrieves profiles of the link.

Parameters:

linkId Identifier of the link.

type Type of profiles to be retrieved. If AH_PROFILE_UNDEF type is used, all kinds of profiles will be taken.

size Pointer to array size. Initially, the variable must contain capacity of the output array. On output, it will contain how many members of output array were filled.

prof Pointer to array to be filled with link profiles. If this pointer is NULL, *size will be set to size of array necessary to hold all the available data.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_MORE Function was successful; more data is available.

AH_WARN_LINK_EDITED Function was successful; new link data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERR_LINK_DISCARDED Link is no more available.

AH_ERROR_CODE AH_GetLinkConnections
(IN AH_LINK_ID *linkId*,
INOUT AH_COUNTER * *size*,
OUT AH_LINK_CONNECTION * *conn*)

Retrieves connections on the end of the link.

Parameters:

linkId Identifier of the link.

size Pointer to array size. Initially, the variable must contain capacity of the output array. On output, it will contain how many members of output array were filled.

conn Pointer to array to be filled with link connections. If this pointer is NULL, *size will be set to size of array necessary to hold all the available data.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_MORE Function was successful; more data is available.

AH_WARN_LINK_EDITED Function was successful; new link data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERR_LINK_DISCARDED Link is no more available.

AH_ERROR_CODE AH_GetLinkConnection
(IN AH_LINK_ID *linkId*,
OUT AH_LINK_CONNECTION * *conn*)

Retrieves most probable connection on the end of the link.

Parameters:

linkId Identifier of the link.

conn Pointer of structure to be filled with most probable link connection.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_LINK_EDITED Function was successful; new link data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERR_LINK_DISCARDED Link is no more available.

AH_ERROR_CODE AH_GetLinkPoints
(IN AH_LINK_ID *linkId*,
INOUT AH_COUNTER * *size*,
INOUT AH_GEOPOINT * *pt*)

Retrieves geometry of the link.

Parameters:

linkId Identifier of the link.

size Pointer to array size. Initially, the variable must contain capacity of the output array. On output, it will contain how many members of output array were filled.

pt Pointer to array to be filled with link shape points. If this pointer is NULL, *size will be set to size of array necessary to hold all the available data.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_MORE Function was successful; more data is available.

AH_WARN_LINK_EDITED Function was successful; new link data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERR_LINK_DISCARDED Link is no more available.

AH_ERROR_CODE AH_GetPathLength
(IN AH_PATH_INDEX *pathIndex*,
OUT AH_DISTANCE * *length*)

Retrieves length of the complete path. Currently, only pathIndex = 0 (most probable path) is supported. Please note that path length is sum of all links belonging to the path. Current position on the path is not taken in account.

Parameters:

pathIndex Index of the path.

length Pointer to variable that will receive length of the path. Length will be expressed in current distance units.

Returns:

AH_SUCCESS Function was successful.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERROR_CODE AH_GetPathOnRoute
(IN AH_PATH_INDEX *pathIndex*,
OUT AH_BOOL * *onRoute*)

Checks if complete path is part of the calculated route. Currently, only pathIndex=0 (most probable path) is supported.

Parameters:

pathIndex Index of the path.

onRoute Pointer to variable that will receive on-route flag.

Returns:

AH_SUCCESS Function was successful.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERROR_CODE AH_GetPathAttachments
(IN AH_PATH_INDEX *pathIndex*,
IN AH_ATTACHMENT_TYPE *type*,
INOUT AH_COUNTER * *size*,
OUT AH_ATTACHMENT * *att*)

Retrieves attachments of the path. Currently, only pathIndex = 0 (most probable path) is supported.

Parameters:

pathIndex Index of the path.

type Type of attachment to be retrieved. If AH_LA_TYPE_UNDEF is used, all attachments will be taken.

size Pointer to array size. Initially, the variable must contain capacity of the output array. On output, it will contain information, how many members of output array were filled.

att Pointer to array to be filled with path attachments. If this pointer is NULL, *size will be set to estimated size of array necessary to hold all the available data. Estimated size may be greater than actual number of attachments that will be returned.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_MORE Function was successful; more data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERROR_CODE AH_GetPathProfiles
(IN AH_PATH_INDEX *pathIndex*,
IN AH_PROFILE_TYPE *type*,
INOUT AH_COUNTER * *size*,
OUT AH_PROFILE * *prof*)

Retrieves profiles of the path. Currently, only pathIndex = 0 (most probable path) is supported.

Parameters:

pathIndex Index of the path.

type Type of profiles to be retrieved. If it is set to AH_PROFILE_UNDEF , all kinds of profiles will be taken.

size Pointer to array size. Initially, the variable must contain capacity of the output array. On output, it will contain the information, how many members of output array were filled.

prof Pointer to array to be filled with path profiles. If this pointer is NULL, *size will be set to recommended size of the array necessary to hold all the available data. Real number of profiles returned may be smaller than estimated size.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_MORE Function was successful; more data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERROR_CODE AH_GetPathConnections
(IN AH_PATH_INDEX *pathIndex*,
INOUT AH_COUNTER * *size*,
OUT AH_LINK_CONNECTION * *conn*,
OUT AH_DISTANCE * *dist*)

Retrieves connections along the path. Currently, only pathIndex = 0 (most probable path) is supported. Connections between links on the path will not be returned!

Parameters:

pathIndex Index of the path.

size Pointer to array size. Initially, the variable must contain capacity of the output array. On output, it will contain how many members of output array were filled.

conn Pointer to array to be filled with link connections. If this pointer is NULL, *size will be set to size of array necessary to hold all the available data.

dist Pointer to array to be filled with distances from start of the path to connections. Distances will be expressed in current distance units. If this pointer is NULL, *size will be set to size of array necessary to hold all the available data.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_MORE Function was successful; more data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

AH_ERROR_CODE AH_GetPathPoints
(IN AH_PATH_INDEX *pathIndex*,
INOUT AH_COUNTER * *size*,
OUT AH_GEOPOINT * *pt*)

Retrieves geometry of the path. Currently, only pathIndex = 0 (most probable path) is supported.

Parameters:

pathIndex Index of the path.

size Pointer to array size. Initially, the variable must contain capacity of the output array. On output, it will contain information, how many members of output array were filled.

pt Pointer to array to be filled with path shape points. First point is same as current position. If this pointer is NULL, *size will be set to size of array necessary to hold all the available data.

Returns:

AH_SUCCESS Function was successful.

AH_WARN_MORE Function was successful; more data is available.

AH_ERR_UNDEFINED The information is unavailable.

AH_ERR_INVALID_PARAMETER One or more parameters are invalid.

2.3. Positioner

2.3.1. Positioner enumerations

enum AH_INTEGRITY_GNSS
enum AH_INTEGRITY_DR
enum AH_INTEGRITY_MAP

Enumeration Type Documentation

enum AH_INTEGRITY_GNSS

This enumeration specifies the quality of estimation the position, when the GNSS system only was used.

Enumerator:

AH_INTEGRITY_GNSS_1 Position was determined with a GNSS - 3D fixed (4 or more satellites in good constellation) for at least the last 60 seconds. Hence latitude, longitude, and altitude can be determined correctly.

AH_INTEGRITY_GNSS_2 Position was determined with at least a GNSS - 2D fixed for at least the last 60 seconds (including perhaps for some period a 3D fix, but not over the complete period). Hence latitude and longitude can be determined approximately, using assumed altitude, and a possibility exists for inaccuracies due to wrong altitude value.

AH_INTEGRITY_GNSS_3 Position was determined without a GNSS fix for some period in the last 60 seconds.

AH_INTEGRITY_GNSS_UNDEF No information about the usage of GNSS is available.

enum AH_INTEGRITY_DR

This enumeration specifies the quality of estimation the position, when the DR system was used.

Enumerator:

AH_INTEGRITY_DR_1 Position was determined with aligned GNSS fix and dead reckoning for at least the last 60 seconds, GNSS and dead-reckoning agree on position, no reflections in GNSS /drift in DR sensors influences

AH_INTEGRITY_DR_2 Position was determined with dead reckoning for at least the last 60 seconds, but GNSS fix and dead reckoning where not always aligned in last 60 seconds, Due to reflections/GNSS intermittent failure of reception/drift in sensors, disconcordance dead-reckoning estimate and GNSS fix occurred.

AH_INTEGRITY_DR_3 Position was determined without dead reckoning for some period in the last 60 seconds

AH_INTEGRITY_DR_UNDEFINED No information about the usage of DR is available.

enum AH_INTEGRITY_MAP

This enumeration specifies how a digital map was used to specify position of the vehicle.

Enumerator:

AH_INTEGRITY_MAP_1 Position is in fully digitized map area.

AH_INTEGRITY_MAP_2 Position is in partly digitized map area.

AH_INTEGRITY_MAP_3 Position is outside digitized map area.

AH_INTEGRITY_MAP_UNDEF No information about the usage of map is available.

2.3.2. Positioner structures

struct AH_GEOPOINT
struct AH_POSITION_EST
struct AH_POSITION_MATCHED
struct AH_POSITION_EXTRAPOLATED

2.3.3. Positioner functions

AH_DISTANCE AH_DistanceTo
(IN const AH_GEOPOINT *point)
AH_DISTANCE AH_DistanceDriveTo
(IN const AH_GEOPOINT *point)
AH_ERROR_CODE AH_GetMPosAlternatives
(IN AH_POSITION_MATCHED *pos,
IN AH_COUNTER *num)
AH_ERROR_CODE AH_GetPositionMatched
(OUT AH_POSITION_MATCHED *position)
AH_ERROR_CODE AH_GetPositionEstimated
(OUT AH_POSITION_EST *position)
AH_ERROR_CODE AH_GetPositionExtrapolated
(OUT AH_POSITION_EXTRAPOLATED *position)

Function Documentation

AH_DISTANCE AH_DistanceTo
(IN const AH_GEOPOINT * *point*)

Gets air distance (shortest spherical distance) from the current position to the defined point. Calculations are performed in the 2-dimensional space; the altitude is not taken into account.

Parameters:

point Pointer to the AH_GEOPOINT structure containing description of the referenced point.

Returns:

Returns the distance to the defined point in meters with fractions. Distance is defined according the rules of spherical geometry.

Return values:

AH_DISTANCE_UNDEF if distance or input pointer is not valid

AH_DISTANCE AH_DistanceDriveTo
(IN const AH_GEOPOINT * *point*)

Gets drivable distance from the current position to the defined point. Calculations are performed in the 2-dimensional space; the altitude is not taken into account. Distance is calculated along the drivable way to the defined point.

Parameters:

point Pointer to the AH_GEOPOINT structure containing description of the referenced point.

Returns:

Returns the distance to the defined point in meters with fractions. Distance is defined according the rules of spherical geometry.

Return values:

AH_DISTANCE_UNDEF if distance or input pointer is not valid

AH_ERROR_CODE AH_GetMPosAlternatives
(IN AH_POSITION_MATCHED * *pos*,
IN AH_COUNTER * *num*)

Allows getting the array of alternative matched positions (except the most probable one).

Parameters:

pos Pointer to the array of AH_POSITION_MATCHED structures containing alternatively matched position.

num Pointer to the number of alternative position to be taken.

Precondition:

If NULL pointer is passed to the function, the number of existing alternative position will be returned.

Postcondition:

After return from the function, the variable contains the number of elements actually placed into the list, if code is AH_SUCCESS; otherwise it is the NULL pointer.

Returns:

AH_SUCCESS code if the operation is successful

AH_ERR_NOT_SUPPORTED in case the current version does not support determination of alternative position(s)

AH_ERR_... one of the error values (dependently of the error reason).

AH_ERROR_CODE AH_GetPositionMatched (OUT AH_POSITION_MATCHED * *position*)

Gets information about matched position corresponding to the current estimated position.

Parameters:

position Pointer to the AH_POSITION_MATCHED structure that receives the description of the matched position.

Returns:

AH_SUCCESS code if the operation is successful.

AH_ERR_... one of the error values (dependently of the error reason).

AH_ERROR_CODE AH_GetPositionEstimated (OUT AH_POSITION_EST * *position*)

Gets information about actual estimated position.

Parameters:

position Pointer to the AH_POSITION_EST structure that receives the description of the estimated position.

Returns:

AH_SUCCESS code if the operation is successful.

AH_ERR_... one of the error values (dependently of the error reason).

AH_ERROR_CODE AH_GetPositionExtrapolated (OUT AH_POSITION_EXTRAPOLATED * *position*)

Gets information about extrapolated position.

Parameters:

position Pointer to the AH_POSITION_EXTRAPOLATED structure that receives the description of the extrapolated position.

Returns:

AH_SUCCESS code if the operation is successful.

AH_ERR_... one of the error values (dependently of the error reason).

2.4. Curve Modeller

2.4.1. Curve Modeller enumerations

enum AH_CURVE_TURN_DIRECTION
enum AH_CURVE_CLUSTER_TYPE

Enumeration Type Documentation

enum AH_CURVE_TURN_DIRECTION

Specifies kinds of curve directions (direction of turn)

Enumerator:

AH_CURVE_TURN_LEFT Direction of curve is left

AH_CURVE_NO_TURN Direct road

AH_CURVE_TURN_RIGHT Direction of curve is right

AH_CURVE_TURN_UNDEF Undefined/invalid direction value

enum AH_CURVE_CLUSTER_TYPE

Specifies methods of clustering for curve elements:

Enumerator:

AH_CLUSTER_TYPE_NO_FILTER Filter is not defined.

AH_CLUSTER_TYPE_RADIUS Curve elements belong to one cluster if radiuses of them not more than the defined value.

AH_CLUSTER_TYPE_RADIUS_DIFF Curve elements belong to one cluster if differences of their radiuses are more than defined value.

AH_CLUSTER_TYPE_RADIUS_DIFF_PERS Curve elements belong to one cluster if differences of their radiuses are more than defined value (in percents to the minimal value).

AH_CLUSTER_TYPE_DIRECTION Curve elements belong to one cluster if they have same curvature sign.

AH_CLUSTER_TYPE_DISTANCE_FROM Curve elements belong to one cluster if distance to the start point of them is less than defined value.

AH_CLUSTER_TYPE_LENGTH Curve elements belong to one cluster if sum of their length is less than defined value.

AH_CURVE_CLUSTER_TYPE_UNDEF Undefined.

2.4.2. Curve Modeller structures

struct AH_CURVE_ELEMENT
struct AH_CURVE_ELEMENT_CLUSTER

2.4.3. Curve Modeller functions

AH_ERROR_CODE AH_GetCurveAt
(IN AH_PATH_INDEX pathIndex,
IN AH_DISTANCE distance,
OUT AH_DISTANCE *radius,
OUT AH_CURVE_TURN_DIRECTION *direction)
AH_ERROR_CODE AH_GetCurveElems
(IN AH_PATH_INDEX pathIndex,
IN AH_COUNTER from,
INOUT AH_COUNTER *n,
OUT AH_CURVE_ELEMENT *elements)
AH_ERROR_CODE AH_GetCurveClusters
(IN AH_PATH_INDEX pathIndex,
IN AH_COUNTER from,
INOUT AH_COUNTER *n,
OUT AH_CURVE_ELEMENT_CLUSTER *elements,
IN AH_CURVE_CLUSTER_TYPE filter_type,
IN AH_INT32 filter_value)
AH_BOOL AH_IsCurve
(IN AH_DISTANCE radius)
AH_BOOL AH_IsCurveFitToReal (void)

Function Documentation

AH_ERROR_CODE AH_GetCurveAt
(IN AH_PATH_INDEX *pathIndex*,
IN AH_DISTANCE *distance*,
OUT AH_DISTANCE * *radius*,
OUT AH_CURVE_TURN_DIRECTION * *direction*)

Provides the radius and the direction of the curvature at the requested position around the current position.

Parameters:

pathIndex Index of the path, for which the operation should be performed

distance Defines the relative position in front of the current map-matched position, where curvature is requested. If distance is negative, the curvature at the point behind the current position is returned.

radius The radius at the requested point on route.

direction The direction of the curvature.

Returns:

AH_SUCCESS if the routine returns successfully.

AH_ERR_OVER_RANGE if distance is longer than the currently estimated main-path.

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_GetCurveElems
(IN AH_PATH_INDEX *pathIndex*,
IN AH_COUNTER *from*,
INOUT AH_COUNTER * *n*,
OUT AH_CURVE_ELEMENT * *elements*)

For the current curve gets a sub-list of AH_CURVE_ ELEMENTS in front of the vehicle with up to /i n elements (or through the end of the curvatures sequence) beginning at the specified number.

Parameters:

pathIndex Index of the path, for which the operation should be performed

from number of the first element in the list, which should be taken

n Pointer to the variable, containing the number of element to be taken, if n is equal to 0, all elements until end of the list should be taken. After return from the function, the variable contains the number of elements actually placed into the list, if code is AH_SUCCESS, otherwise it is NULL pointer.

elements Pointer to the array of AH_CURVE_ELEMENT structures

Returns:

AH_SUCCESS if the routine returns successfully.

AH_ERR_NOT_SUPPORTED in case the current version of Horizon does not support curvature model,

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_ERROR_CODE AH_GetCurveClusters
(IN AH_PATH_INDEX *pathIndex*,
IN AH_COUNTER *from*,
INOUT AH_COUNTER * *n*,
OUT AH_CURVE_ELEMENT_CLUSTER * *elements*,
IN AH_CURVE_CLUSTER_TYPE *filter_type*,
IN AH_INT32 *filter_value*)

Gets a sub-list of AH_CURVE_ ELEMENTS_CLUSTER in front of the vehicle with up to *n* elements (or through the end of the curvatures sequence) beginning at the specified number. Elements are grouped into the cluster correspondently to the filter_type and filter_value. Filter type can be one AH_CURVE_CLUSTER_TYPE enumeration value.

Parameters:

pathIndex Index of the path, for which the operation should be performed.

from Number of the first curve element in the curve, where clustering should start.

n Pointer to the variable containing the maximal number of cluster that should be formed. After returns from the function, the variable contains the number of clusters actually placed into the list, if code is AH_SUCCESS, otherwise it is NULL pointer.

elements Pointer to the array of AH_CURVE_ELEMENT_CLUSTER structures.

filter_type Type of the filter that determines the clusterisation procedure. See AH_CURVE_CLUSTER_TYPE enumeration for details.

filter_value Filter parameter as integer value used to define boundaries of clusters.

Returns:

AH_SUCCESS if the routine returns successfully.

AH_ERR_NOT_SUPPORTED in case the current version of Horizon does not provide clusterization,

AH_ERR_... one of the error values (dependently of the error reason) otherwise.

AH_BOOL AH_IsCurve (IN AH_DISTANCE *radius*)

Checks whether some element with radius less than defined threshold exists along the current curve. Function returns AH_FALSE if all elements of the curve have radiuses more than this value.

Parameters:

radius Maximal threshold of the radius. If the radius of a segment is more than this value, segment should be considered as a straight line.

Returns:

AH_TRUE if some curve elements can be defined in front of the vehicle,

AH_FALSE otherwise

AH_BOOL AH_IsCurveFitToReal (void)

Checks whether the real parameters of vehicle movement (curve radius calculated via the angle rate) correspond to the curve parameters provided by the electronic map.

Returns:

AH_TRUE in case parameters of movements correspond to the curvature definition,

AH_FALSE otherwise

3. API Data Structure Documentation

3.1. AH_ATTACHMENT Structure

3.1.1. Detailed Description

The AH_ATTACHMENT structure defines the properties of an attachment delivered by the AHP.

3.1.2. Data Fields

AH_DISTANCE dist
AH_ATTACHMENT_TYPE type
union {
  AH_DESCRIPTION description
  AH_SIGN sign
} X

3.1.3. Field Documentation

AH_DISTANCE dist

Distance of the attachment from start of the link (if retrieved using AH_GetLinkAttachments()), or from the start of the path (if retrieved using AH_GetPathAttachments()).

AH_ATTACHMENT_TYPE type

Type of the attachment. See AH_ATTACHMENT_TYPE enumeration for possible value.

AH_DESCRIPTION description

Full description of the attachment if type is not equal to AH_LA_TRAFFIC_SIGN . See AH_DESCRIPTION structure.

AH_SIGN sign

Description of traffic sign if type is equal to AH_LA_TRAFFIC_SIGN . See AH_SIGN structure.

union { ... } X

Union to save space.

3.2. AH_CURVE_ELEMENT Structure

3.2.1. Detailed Description

Defines parameters of the curve segment.

3.2.2. Data Fields

AH_LINK_ID link_Id
AH_GEOPOINT start_point
AH_DISTANCE distance_to
AH_DISTANCE length
AH_DISTANCE radius
AH_INT8 coefficient
AH_CURVE_TURN_DIRECTION direction

3.2.3. Field Documentation

AH_LINK_ID link_Id

Id of the link, where curve element is started

AH_GEOPOINT start_point

Start point of the curvature segment

AH_DISTANCE distance_to

Distance (driveable) from the beginning of the curvature (matched position) to the segment.

AH_DISTANCE length

Length of the curvature segment in meters. Length is calculated as sum of sphere distances between curvature vertexes belonging to this segment.

AH_DISTANCE radius

Radius of the curvature in meters at the beginning of the curve element.

AH_INT8 coefficient

Linear coefficient of increasing/decreasing the radius along the curve element, in case the curve element is the part of a circle, this parameter is equal to zero.

AH_CURVE_TURN_DIRECTION direction

Parameter that define position of curvature center relative to the curvature. Can be one of the AH_CURVE_TURN_DIRECTION enumeration value.

3.3. AH_CURVE_ELEMENT_CLUSTER Structure

3.3.1. Detailed Description

Defines cluster of the curve elements.

3.3.2. Data Fields

AH_PATH_INDEX path_index
AH_GEOPOINT start_point
AH_DISTANCE length
AH_DISTANCE radius_min
AH_DISTANCE radius_max
AH_DISTANCE radius_avr

3.3.3. Field Documentation

AH_PATH_INDEX path_index

Index of the path, where the curve is placed.

AH_GEOPOINT start_point

Start point of the cluster of curvature segments

AH_DISTANCE length

Sum of curve element lengths.

AH_DISTANCE radius_min

Minimal radius of the curvature in the cluster.

AH_DISTANCE radius_max

Maximal radius of the curvature in the cluster.

AH_DISTANCE radius_avr

Average radius of the curvature in the cluster.

3.4. AH_DESCRIPTION Structure

3.4.1. Detailed Description

This structure provides a description of the single item (link attachment or profile) associated with the path.

3.4.2. Data Fields

AH_INT32 value
AH_ACCURACY accuracy

3.4.3. Field Documentation

AH_INT32 value

Value of the item. For every type of the link attachment or profile the own enumeration is used.

AH_ACCURACY accuracy

Accuracy of the provided value. One of the 4 possible predefined values.

3.5. AH_GEOPOINT Structure

3.5.1. Detailed Description

The structure defines the position in the Geographic coordinates system.

3.5.2. Data Fields

AH_COORDINATE latitude
AH_COORDINATE longitude

3.5.3. Field Documentation

AH_COORDINATE latitude

Absolute coordinate, latitude according to WGS84.

AH_COORDINATE longitude

Absolute coordinate, longitude according to WGS84.

3.6. AH_LINK_CONNECTION Structure

3.6.1. Detailed Description

The AH_LINK_CONNECTION structure defines the properties of link connection.

3.6.2. Data Fields

AH_LINK_ID link_id_to
AH_TRANSITION transition_type
AH_PROBABILITY probability
AH_BOOL u_turn
AH_HEADING heading_change
AH_JUNCTION_TYPE junction
AH_ACCURACY accuracy
AH_GIVEWAY giveway

3.6.3. Field Documentation

AH_LINK_ID link_id_to

For connected links: the unique identifier of the second link in the pair.

AH_TRANSITION transition_type

Description of the transition at the connection (a crossing) represented by one of the AH_TRANSITION enumeration values (it is possible and/or allowed to drive to the street represented by / i link_to). See AH_TRANSITION for details.

AH_PROBABILITY probability

Probability to drive from current link to the link specified by /i link_id_to.

AH_BOOL u_turn

Flag determining is it possible to make u-turn at this connection.

AH_HEADING heading_change

Angle between current link and link defined by /i link_id_to.

AH_JUNCTION_TYPE junction

The classification of a junction (see AH_JUNCTION_TYPE enumeration)

AH_ACCURACY accuracy

Accuracy of the provided value. One of the 4 possible predefined values (see "Interface and Data Entity Specifications. Part 1 - General Specifications" document, Chapter 4.2.2.).

AH_GIVEWAY giveway

Right of way regulation

3.7. AH_METADATA Structure

3.7.1. Detailed Description

Metadata

3.7.2. Data Fields

AH_PROVIDERS_MAP map_provider_ID
AH_PROVIDERS_HORIZON horizon_provider_ID
AH_UINT8 country_codes_num
const AH_CHAR * country_codes
AH_CHAR version [AH_VERSION_LENGTH]

3.7.3. Field Documentation

AH_PROVIDERS_MAP map_provider_ID

Unique map provider identification. Specified by one of the AH_PROVIDERS_MAP enumeration value.

AH_PROVIDERS_HORIZON horizon_provider_ID

Unique ADAS horizon provider identification. Specified by one of the AH_PROVIDERS_HORIZON enumeration value.

AH_UINT8 country_codes_num

Number of country codes in the map coverage zone

const AH_CHAR* country_codes

Array of country codes (length of each code is 3 characters, values correspond to the ISO 3166 alpha-3 Standard) which represent the coverage zone of the current map. Null-terminated string is used.

AH_CHAR version[AH_VERSION_LENGTH]

Version of the ADAS horizon and protocol.

3.8. AH_METADATA_EXT Structure

3.8.1. Detailed Description

The extended metadata include additional information which is only delivered on request of the ADAS application:

3.8.2. Data Fields

AH_DRIVING_SIDE driving_side
AH_LENGTH_UNITS length_units
AH_SP_DELTA_UNITS sp_delta_units
AH_TIME_ZONE time_zone
const AH_CHAR * horizon_supplier

3.8.3. Field Documentation

AH_DRIVING_SIDE driving_side

Default driving side for countries included into the map

AH_LENGTH_UNITS length_units

Length units used in the system (see AH_LENGTH_UNIT enumeration)

AH_SP_DELTA_UNITS sp_delta_units

Defines the units (meters or degrees) used for transmitting deltas between shape points

Note:

Version 1.0 supports degrees only. Usage of degree can be dangerous in case the system should transmit high latitude and longitude values.

AH_TIME_ZONE time_zone

Time zone determines the difference in hours between the time on the host computer and Universal Coordinated Time (UTC). One of AH_TIME_ZONE enumeration values is used.

const AH_CHAR* horizon_supplier

Name of ADAS horizon supplier

3.9. AH_POSITION_EST Structure

3.9.1. Detailed Description

This structure defines the current (estimated) position of a vehicle.

3.9.2. Data Fields

AH_GEOPOINT coordinates
AH_UINT32 integrity
AH_TIME time
AH_HEADING heading
AH_SPEED speed

3.9.3. Field Documentation

AH_GEOPOINT coordinates

Absolute coordinates, latitude and longitude, according to WGS84.

AH_UINT32 integrity

A measure of trust, which can be placed in the correctness of the positioning information supplied by the total system. Positioning integrity includes the ability of the system to provide timely warnings to the user when the system should not be used for the intended operation. Defined by the combination of AH_INTEGRITY_GNSS and AH_INTEGRITY_DR integrity indicators.

AH_TIME time

Dependent on the AH_METADATA value, one of two possible values: timestamp or age, where: Timestamp is the moment in time (relative value in the range from 0 to 4095), when the vehicle was expected to be at the estimated position; Age is the time difference between the time-stamp for which this position is determined and the current time. Note that if the position has been extrapolated the age can be set to 0 even if it is based on sensor data that was captured some time ago.

AH_HEADING heading

The direction of the vehicle's movement.

AH_SPEED speed

The rate that the vehicle's position is changing in the direction of the vehicle heading.

3.10. AH_POSITION_EXTRAPOLATED Structure

3.10.1. Detailed Description

The structure defines the most probable location of the vehicle by taking into account the time delay after the last position message was received from the Provider side.

3.10.2. Data Fields

AH_LINK_ID link_id
AH_GEOPOINT coordinates
AH_DISTANCE offset

3.10.3. Field Documentation

AH_LINK_ID link_id

Unique identifier of the link.

AH_GEOPOINT coordinates

Absolute coordinates, latitude and longitude according to WGS84 of the extrapolated position.

See also:

AH_GEOPOINT.

AH_DISTANCE offset

Distance from the estimated position.

3.11. AH_POSITION_MATCHED Structure

3.11.1. Detailed Description

This structure defines the most probable location of the vehicle on a part of the road network.

3.11.2. Data Fields

AH_LINK_ID link_id
AH_GEOPOINT coordinates
AH_DISTANCE offset
AH_UINT32 integrity
AH_TIME time
AH_HEADING heading
AH_SPEED speed

3.11.3. Field Documentation

AH_LINK_ID link_id

Unique identifier of the link.

AH_GEOPOINT coordinates

Absolute coordinates, latitude and longitude according to WGS84 of the position matched to the road network.

See also:

AH_GEOPOINT.

AH_DISTANCE offset

Distance from the estimated position.

AH_UINT32 integrity

A measure of trust, which can be placed in the correctness of the positioning information supplied by the total system. Positioning integrity includes the ability of the system to provide timely warnings to the user, when the system should not be used for the intended operation. Defined by the combination of AH_INTEGRITY_GNSS , AH_INTEGRITY_DR and AH_INTEGRITY_MAP indicators described below.

AH_TIME time

Dependent on the AH_METADATA value, one of two possible values: timestamp or age, where: Timestamp is the moment in time (relative value in the range from 0 to 4095), when the vehicle was expected to be at the estimated position; Age is the time difference between the time-stamp for which this position is determined and the current time. Note that if the position has been extrapolated the age can be set to 0 even if it is based on sensor data that was captured some time ago.

AH_HEADING heading

The direction of the vehicle's movement relative to the road segment.

AH_SPEED speed

The rate that the vehicle's position is changing in the direction of the vehicle heading movement projected on a road.

3.12. AH_PROFILE Structure

3.12.1. Detailed Description

The AH_PROFILE structure defines the properties of a profile.

3.12.2. Data Fields

AH_DISTANCE dist
AH_PROFILE_TYPE type
AH_DESCRIPTION description

3.12.3. Field Documentation

AH_DISTANCE dist

Distance of the profile from start of the link (if retrieved using AH_GetPathProfiles()), or from the start of the path (if retrieved using AH_GetPathProfiles()).

AH_PROFILE_TYPE type

Type of the profile. See AH_PROFILE_TYPE enumeration for possible value.

AH_DESCRIPTION description

Full description of the profile. See AH_DESCRIPTION structure.

3.13. AH_PROPERTIES_MAIN Structure

3.13.1. Detailed Description

The structure defines other parameters, which are important for the system.

3.13.2. Data Fields

AH_LENGTH_UNITS length_units
AH_DISTANCE reference_max_distance
AH_TIME reference_max_interval
AH_ACCURACY accuracy
AH_BOOL interactive

3.13.3. Field Documentation

AH_LENGTH_UNITS length_units

Resolution of the lengths and distances measurement. Defined by one of the AH_LENGTH_UNITS enumeration's value

AH_DISTANCE reference_max_distance

Maximal distance, which can be used as relative distance from a reference point.

AH_TIME reference_max_interval

Maximal time interval, within that the reference point should be updated.

AH_ACCURACY accuracy

Accuracy of incoming data and calculations. One of the values of the enumeration AH_ACCURACY, which represents accuracy categories.

AH_BOOL interactive

Defines, whether the bi-directional communication between Provider and Reconstructor exists.

3.14. AH_SETTINGS_MODELLER Structure

3.14.1. Detailed Description

Settings for the Curve Modeller component

3.14.2. Data Fields

AH_MODEL_CURVE model
AH_BOOL curve_clusters_supported
AH_UINT32 max_curve_radius

3.14.3. Field Documentation

AH_MODEL_CURVE model

Determines the type of model. Defined by one of the AH_MODEL_CURVE enumeration's value

AH_BOOL curve_clusters_supported

Defines, whether the curve clusters are supported in the current version of Horizon or not.

AH_UINT32 max_curve_radius

Determines maximal radius of curvature that is taken into account by Horizon. If this threshold is exceeded, the curve segment is considered as the straight line.

3.15. AH_SETTINGS_POSITIONER Structure

3.15.1. Detailed Description

Settings for the Positioner component

3.15.2. Data Fields

AH_COUNTER mm_pos_num
AH_TIME min_mm_pos_interval
AH_TIME max_est_pos_interval
AH_TIME max_mm_pos_interval

3.15.3. Field Documentation

AH_COUNTER mm_pos_num

Maximal number of map_matched positions that can be received from the Horizon.

AH_TIME min_mm_pos_interval

Minimal interval, within which the matched position should be updated.

AH_TIME max_est_pos_interval

Maximal interval, within which the estimated position should be updated

AH_TIME max_mm_pos_interval

Maximal interval, within which the map_matched position should be updated

3.16. AH_SETTINGS_RECONSTRUCTOR Structure

3.16.1. Detailed Description

The structure controls attributes of the Path Reconstructor component.

3.16.2. Data Fields

AH_MODEL_PATH model
AH_DISTANCE max_length
AH_DISTANCE min_stubs_length
AH_DISTANCE max_stubs_length
AH_BOOL attachment_options [AH_LA_TYPE_CNT]
AH_BOOL profile_options [AH_PROFILE_TYPE_CNT]

3.16.3. Field Documentation

AH_MODEL_PATH model

Type of the Path Reconstructor model. Defined by one of the AH_MODEL_PATH enumeration's value

AH_DISTANCE max_length

Maximal length of the Horizon (Maximal length of a single path) in meters.

AH_DISTANCE min_stubs_length

Minimal length of stubs (in meters), which should be provided by the Horizon.

AH_DISTANCE max_stubs_length

Maximal length of stubs (in meters), which should be provided by the Horizon.

AH_BOOL attachment_options[AH_LA_TYPE_CNT]

Array of boolean value that determine, which link attachment types are provided by the Horizon. Order of values in the array corresponds to the AH_ATTACHMENT_TYPE structure.

AH_BOOL profile_options[AH_PROFILE_TYPE_CNT]

Array of boolean value that determine, which profile types are provided by the Horizon. Order of values in the array corresponds to the AH_PROFILE_TYPE enumeration.

3.17. AH_SIGN Structure

3.17.1. Detailed Description

The AH_SIGN structure provides a description of a traffic sign.

3.17.2. Data Fields

AH_SIGN_LOCATION location
AH_SIGN_TYPE type
AH_DISTANCE validity
AH_UINT16 value

3.17.3. Field Documentation

AH_SIGN_LOCATION location

Location of the sign relative to the road. Can be one of the AH_SIGN_LOCATION enumeration values.

AH_SIGN_TYPE type

Sign type (category). Can be one of the AH_SIGN_TYPE enumeration values.

AH_DISTANCE validity

Distance of the sign validity.

AH_UINT16 value

Meaning of the sign. Sign is defined by integer value which represent its official number.