ADASIS AISBL

ADASIS v3 Protocol

Specification v3.1.0

October 01 2018

Proprietary and Confidential

Document Control

Document title	ADASIS v3 Protocol
Electronic name	ADASIS v3 Protocol v3.1.0.pdf
Main author(s) or editor(s)	
Current author(s)	
Contributing author(s)	

Copyright notes

Document history

Version	3.1.0
Status	
Date	2018-10-01
Authors	
Reviewers	
Summary of Changes	Initial Release with: - comments in section 1.1.1 regarding next protocol update to 3.2.0 - Changing from ADASIS Forum to the legal form ADASIS AISBL

Abstract

The ADASIS AISBL seeks to standardize the interface between a digital map and ADAS applications. This interface predictively provides ADAS applications with road characteristics (e.g., road/lane geometry, slope, speed limits, lane markings, etc.) ahead of a vehicle based on the vehicle's position. This concept is referred to as *ADAS Horizon*.

As a vehicle's route or destination can change, the ADAS horizon flexibly adapts and has also the capability to convey dynamic information (e.g., traffic lights status). Further, the ADAS Horizon is able to incorporate auxiliary data sources (e.g., traffic information) and support sensor fusion as well as various communication schemes.

List of Figures

Glossary of Terms

ADAS	Advanced Driver Assistance Systems
ADASIS	ADAS Interface Specification
ADASIS AISBL	ADASIS legal form as a non-profit international association under Belgium law (AISBL= Association Internationale Sans But Lucratif)
Av2	Version 2 of the ADASIS protocol definition (to distinguish from ADASIS version 1 that has been developed and tested in the Prevent Maps & ADAS project)
Av3	Version 3 of the ADASIS protocol definition (to distinguish from Av2)
CAN (Controller Area Network)	A message based protocol used in automotive applications to allow devices to communicate
CAN-FD (CAN with flexible data rate)	CAN FD is an extension of the CAN protocol. It allows messages to hold up to 64 bytes.
ADAS Horizon (AH)	Extract of the Map Database in front of the vehicle's current position
Horizon Provider (Av3HP)	The software component that generates the Av3 ADAS Horizon for transmission to other applications
Horizon Reconstructor (Av3HR)	Implementation of the Av3 Programming Interface that rebuilds the Horizon produced by the Av3HP
Network of Links	A collection of links, connected at their end points, forming the road network
Path	A possible route through the road network that a vehicle could follow
Profile	Description of an attribute of the road network along a path
Profile Entry	Description of an attribute of the road network at a given location on a path
Profile Type	Type of profile information, e.g. curvature or slope
Segment	Part of a Path where the attributes, most significant for ADAS applications, remain the same
Link	An underlying map database component representing the roadway between two intersections
Most Preferred Path (MPP, formerly Most Probable Path)	The most likely route of the vehicle on the road, which may be composed of parts of multiple other paths
Root Path	A path that does not have a parent
Sub Path	A path branching off from a parent path
Parent Path	Path to which a sub path is attached
Vehicle Path	Path that the vehicle currently is located on

Transmitted Path	Path that has been actually transmitted from the Provider to the Client via the network
Path Identifier	Unique identification number of a Path
Offset	Position along a path
Unknown Value	The value assigned to an attribute that is supported by the Horizon Provider but is not available at the current location or data item
Signal not Available Value (N/A)	The value assigned to an attribute that is not supported or not implemented by the current Horizon Provider
Spline	A representation of a curve by polynomial functions based on a set of control points
Clothoid	A curve whose curvature increases or decreases linearly with distance and describes the optimal form of a curve in the road network
Message Type	Type of a message in the scope of the ADASIS v3 protocol
Path Control Message	Path Control Message
Profile Control Message	Profile Control Message

Table of Contents

1 Introduction

The potential of digital map data, normally used only by a navigation system, is of greater purpose if the map data is made accessible for other in-vehicle applications. An increasing number of vehicle functions can make use of information on the road ahead. In contrast to a camera or a radar system, digital maps can provide additional information of a much greater range and of predictive character. Further, the map data can complement the information provided by radar, camera, ultrasound, or LiDaR sensors.

1.1 ADASIS v3 Specification

ADASIS (Advanced Driver Assistance Systems Interface Specification) defines the concept of an “ADAS Horizon” as a means to convey the part of the road network and its characteristics originating from map data along the road ahead of the vehicle. To achieve this, an “ADAS Horizon” is constructed using the current vehicle position and possible paths through the road network. This results in a hierarchy of ways through the network of roads. Furthermore, road geometry is derived and related attributes are added to the vehicle’s set of possible paths. This “ADAS Horizon” can be serialized and transmitted over the vehicle communication network.

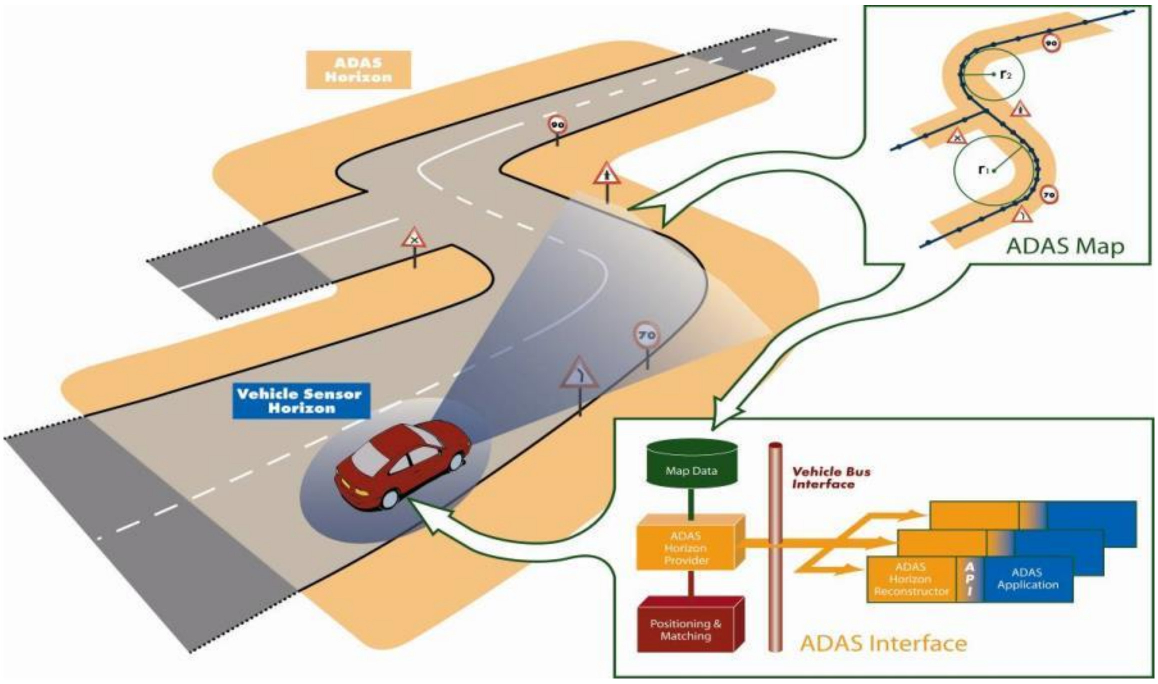


Figure 1 ADAS Concept and Components

The ADASIS AISBL [1] is standardizing the ADASIS protocol for interoperability of different implementations. Based on the experience of the previous versions and the increasing demand for resolution and detail for future applications, the ADASIS v3 specification has been developed.

The purpose of the ADASIS v3 standard is to facilitate the distribution of this ADASIS v3 Horizon data from a horizon provider to clients connected via some communication medium. To achieve this, the ADASIS v3 standard defines data structures for containing this data, using a formal language. Franca IDL was chosen as this formal language. The description file “ADASISv3Messages.fidl” (written in this language) is normative part of the ADASIS v3 standard, together with this document which - besides general and background information - defines the exact semantics of the data structures described in the Franca IDL file.

Thus, the ADASIS v3 standard solely deals with the logical structure of ADASIS v3 Horizon data. In terms of the OSI reference model, it is a layer-7 standard.

At a later point in time, ADASIS may standardize the encoding of ADASIS v3 Horizon data for transmission from a provider to a client (OSI layer 6) and may standardize the actual transmission of ADASIS v3 Horizon data (lower OSI layers). At the time of the creation of this ADASIS v3 standard, though, it just exists non-normative proposals for these layers.

While ADASIS v2 was designed for CAN bus communication restricted to 8 byte messages, ADASIS v3 will make use of higher bandwidth communication means. This allows additional data with more detailed content without the need to use special coding schemes. Messages with a larger payload size facilitate packing a sequence of entity values or to include the representation of a more complex entity in a single payload.

Major improvements of ADASIS v3 are:

- Resolution 0.01 meter instead of 1.0 meter
- Profile entries have an explicit end offset describing the validity or extent of an entry value or the point the next entry is expected on the path representation.
- Multiple profile entries can be packed into one message.
- PathControl, ProfileControl mechanisms
- Dynamic behavior
- Vehicle position message now contains the most probable position and all alternative positions in a single message instead of a set of individual position messages.
- Detailed lane and line geometry
- Additional data (e.g. flow speed)
- Support of detailed information (e.g. HD maps)

The general architecture of systems using ADAS horizon consists of:

- Horizon Provider.
- Horizon Reconstructor.
- ADASIS v3 Protocol.
- ADAS Application.

1.1.1 Outlook and Compatibility with future protocol updates

After release of this specification the development of ADASIS v3 was continued. Next version 3.2.0 will contain new and extended features. This is a short overview:

- Connecting Paths: Connection between tree branches using "merge points" to implement a full network model of the actual road topology.

- New Profiles representing several objects or situations along the road like "Obstacles" for linear geometry objects or "TurnRestriction" to indicate turn restrictions on path level.
- Extension of several enumerations, e.g. "TrafficSigns" which is aligned with several latest national and international standards.

All new concepts and profiles are considered as backwards compatible to 3.1.0 according to the compatibility rules in [2].

The protocol update 3.2.0 will also contain a few specific *not 3.1.0 compatible* changes. These are mainly feature which had not be taken into account or adaptions due to consistency. This is a complete list of incompatible changes:

PROFILE REFERENCES

- RoadAccessibility ProfileValue
  v3.1: Int32ProfileValue -> **v3.2: UInt32ProfileValue**
- FunctionalRoadClass: Introduced value "0" to indicate "unknown"
- Special Situations Profile: Changed interpolation type
  v3.1: step -> **v3.2: spot**

FRANCA IDL REFERENCE

- SignType enumeration: typo fix
  v3.1: 75 = TafficCongestion -> **v3.2: 75 = TrafficCongestion**
- typedef LaneIdx removed and replaced by LaneNumber
  `typedef LaneNumber is UInt8`
  `LaneNumber` is used by
  - Position (Data Type):
    v3.1: `LaneIdx currentLane` -> **v3.2: `LaneNumber`** `currentLane`
  - ProfileEntry (Messages):
    v3.1: `LaneIdx [] laneNumbers` **-> v3.2: `LaneNumber`** `[] laneNumbers`
- DrivingSide (Enumeration): inserted UNKNOWN value
  v3.1: 0 = `RightHandDriving` -> **v3.2: `0 = Unknown, 1 = None, …`**
- LaneTransition (Enumeration): inserted UNKNOWN value
  v3.1: 0 = `None` -> **v3.2: `0 = Unknown, 1 = None, …`**
- LineMarkingColour (Enumeration): inserted UNKNOWN value
  v3.1: 0 = `None` -> **v3.2: `0 = Unknown, 1 = None, …`**
- LocationObjectType (Enumeration): redefinition
  v3.1: 0 = `GuidePost` -> **v3.2: `0 = SignFace, 1 = Pole`**
- RelativeDirection (Enumeration): inserted UNKNOWN value
  v3.1: 0 = `None` -> **v3.2: `0 = Unknown, 1 = None, …`**
- UnitOfSpeed (Enumeration): inserted UNKNOWN value
  v3.1: 0 = `None` -> **v3.2: `0 = Unknown, 1 = KpH, …`**
- LaneConnectivityPair (ProfileValue): introduction of PathID zero
  v3.2: The **incoming side of the main path** is represented by **PathID zero.**

- TrafficLightProfileValue (ProfileValue): struct member `position` replaced
  v3.1: `Vector position` -> **v3.2: `WGS84Point absolutePosition`**

Beside all extensions and changes listed above both documents for ADASIS v3.2 will contain a lot of more explanations. All placeholders and gaps are filled with detailed descriptions and examples.

1.2 The Specification, and the Franca IDL

ADASIS v3 aims at full interoperability between components from different suppliers based on this protocol. To achieve this, it is necessary to define the protocol in detail down to the level of bits and bytes, both syntactically (how is the protocol data structured) and semantically (what is the actual meaning of the protocol data).

The semantics of the protocol generally needs a description in natural language, which can be found in this document. For the syntax, on the other hand, it is possible to use a formal language which gives an unambiguous, machine-readable definition of the protocol data structure.

ADASIS has chosen to employ Franca IDL (see https://github.com/franca/franca) as a tool for a formal definition of the ADASIS v3 protocol.
Franca has been designed as a high-level metalanguage which can be machine-translated into other interface specifications (e.g. Protocol Buffers) or directly into the code. Franca describes data structures and interfaces on a logical level but does not prescribe how these are represented in binary form. For a full specification of data structures on the binary level, a Franca specification needs to be accompanied by a set of translation rules. And data structures still do not make up a communication protocol - in addition there must at least be rules how these data structures are sent over a transport medium. Depending on the medium there may also be the need to define how communication partners can find each other, how they establish communication, etc.

Consequently a full specification of a binary interoperable ADASIS v3 protocol realization consists of multiple parts:

- The Franca IDL file (ADASISv3Messages.fidl) defines the syntactical structure of ADASIS v3 protocol messages.
- This document (the ADASIS v3 protocol specification) both defines the semantics of these messages, and describes how these messages are employed to communicate an ADASIS v3 horizon.
- A translation specification defines how to represent ADASIS v3 messages – as described in Franca – by binary data.
- A transport specification defines how to use a specific lower level transport protocol (e.g. IP) to communicate ADASIS v3 messages.

Currently, only the ADASIS v3 protocol specification (this document) and the Franca IDL file are normative. ADASIS intends to also provide normative translation and transport specifications; and it may be the case that multiple sets of translation and transport specifications might be required to accommodate the needs of different users of ADASIS v3. If this turns out to be the case, that unfortunately would create different ADASIS v3 environments that are not binary compatible, but, on the other hand, it would allow employing ADASIS v3 and even reusing a large part of ADASIS v3 software on systems with mutually incompatible requirements on the transport protocol.

1.2.1 Installing Franca

Actually there's more to Franca than just a formal language. Franca comes with a tool set – based on Eclipse – to edit and to verify specifications written in Franca. This tool set also allows to easily create (using Java) translators that convert Franca into another language – into an interface language like a Protocol Buffers specification, or directly into a programming language like C.

For details how to set up the Franca environment, please consult the Quick Install Guide in the Franca wiki. It is not necessary, though, to use Eclipse Mars as described therein; the current (0.11.1) Franca plug-ins have also been tested to work well with Eclipse Oxygen 3a (4.7.3a) for working with the ADASIS v3 specification. It is expected that later versions of Eclipse will work as well; in any case, it is recommended to start with a DSL Tools package of Eclipse – Franca is such a DSL, and when starting with a standard (i.e. Java developer) Eclipse package, a significant number of tools will need to be installed individually.

On the Franca side, only the Franca Runtime and Franca UI components are needed.

1.2.2 Franca Translators

With the plain Franca environment, it's possible to open, view, and edit the ADASIS v3 Franca specification file, but it is not yet possible to do anything more than that. To go further, one needs to use a translator that converts Franca into another language. The ADASIS AISBL provides, upon request, sample translators to its members (written in Java and Xtend) that translate into Protocol Buffers and into C++. This sample translators are provided as is, without assuring that they are apt for any particular purpose, or for production use. They are intended just as starting point for development of individual target specific translators.

1.2.3 Franca Deployment description

In practice, neither the Franca messages specification file nor the target specific translator will contain the full definition of all targets specific properties of the ADASIS v3 subsystem: Franca offers the option to define configurable properties in a so-called deployment description. This is the right place of configurable parameters such as memory sizes that are not part of the logical ADASIS v3 structure as defined in the Franca messages specification, but still should be easily adaptable and should be managed in a standardized fashion.

Actually there is a deployment definition (in file ADASIS_spec.fdepl) which gives a standardized list of configurable parameters. This file is normative and is provided by the ADASIS AISBL as part of the ADASIS v3 standard. And there is the actual deployment description which gives values for these parameters, in file ADASIS.fdepl. While the ADASIS AISBL provides a sample version of the latter file, it is expected that an OEM defining how to deploy ADASIS v3 in their vehicle will create a customized version suitable for all ADASIS v3 software in this vehicle.

It is expected that suppliers delivering software for this vehicle will use the deployment description provided by the OEM for running the Franca translator and recompiling their software with the result of this translation. This way, all software in the vehicle will use the same OEM specific parameters for the ADASIS v3 protocol, and thus will be binary compatible, without generating disproportionate overhead for OEM specific coding.

Consequently, it is strongly recommended to develop Franca translators in such a way that they make use of a deployment description conforming to the ADASIS v3 standard deployment definition.

1.2.4 Further customization

Beyond defining the values of pre-defined properties for the deployment of ADASIS v3 in their vehicles, it is expected that OEMs also will extend ADASIS v3 by adding custom data, in particular custom profiles. The ADASIS v3 protocol is designed in such a way that this is easily possible, making use of the capabilities of Franca.

The ADASIS AISBL provides a sample illustrating how to do this: The ADASISv3MyCustomized.fidl file contains (commented-out) Franca definitions of a custom profile and of a full custom message type, and the ADASISv3MyCustomized.fdepl file contains a deployment description integrating this with the standard ADASIS v3 data structures. (Unfortunately it is, for technical reasons, necessary to copy the full content of ADASIS.fdepl into ADASISv3MyCustomized.fdepl).

2 ADASIS v3 Concept

2.1 Overview

In a digital map database, real-world objects are represented on different levels of detail.

The simplest representation models roads as lines, and intersections as nodes connecting these lines. The nodes and lines (also called edges or links) together describe the topology of the road network, which is further described by geographic coordinates.

This abstraction of the real road network is shown in Figure 2, where blue lines connect the various nodes and red dots illustrate road shape points. The purple line indicates a branching link.

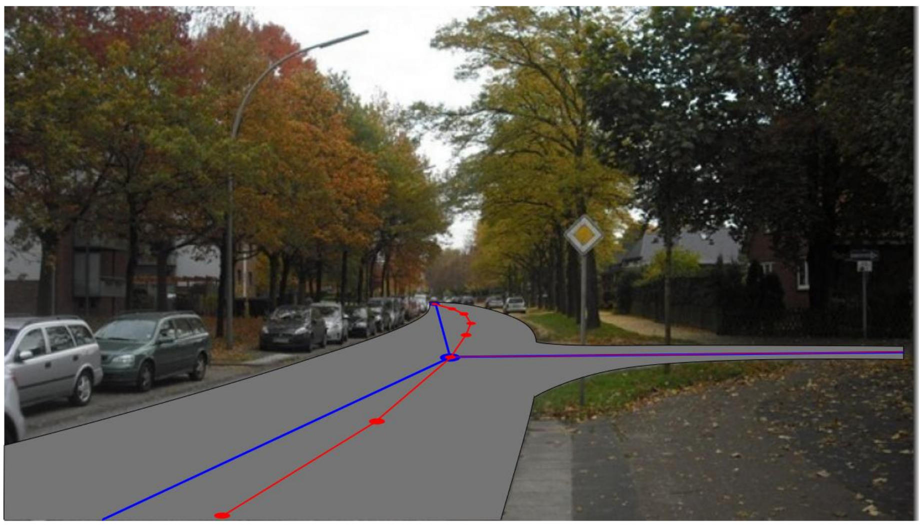


Figure 2 Abstraction of Real World

At this simple T-intersection, the blue circles/lines show the simplified model of the reality. The blue circles represent the nodes of the digital map and the blue lines represent the connectivity between those nodes. For a more realistic representation of the road network attributes like geographical points can be added into the model (red points and lines).

In this way, crossings or other more complex environments can be reduced to a simple abstract view. The above shown example can be described in a very abstract way as it is shown in Figure 3.

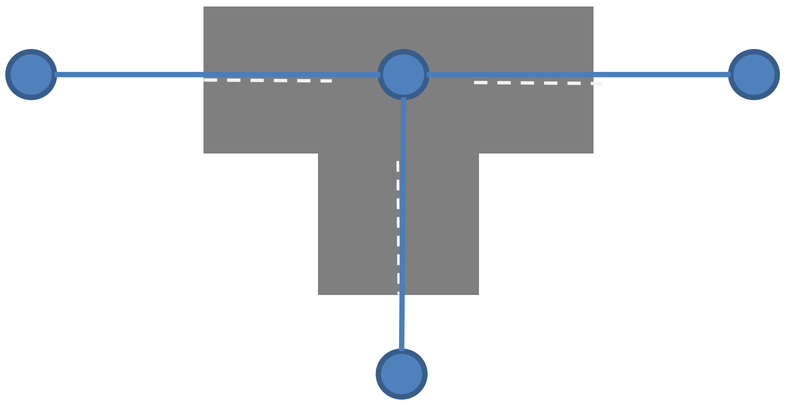


Figure 3 Abstract view of a simple T-intersection

For future applications requiring a higher level of detail, this abstraction of the reality can be used again, but with a more detailed mesh of nodes, which represents a more detailed model.

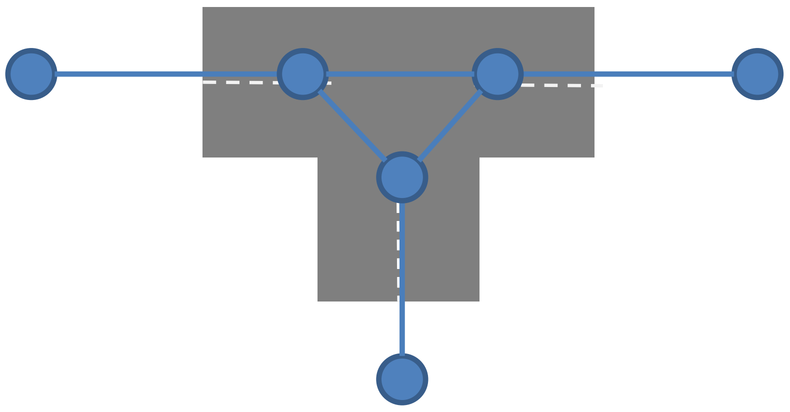


Figure 4 Abstract view of a simple T-intersection in HD maps

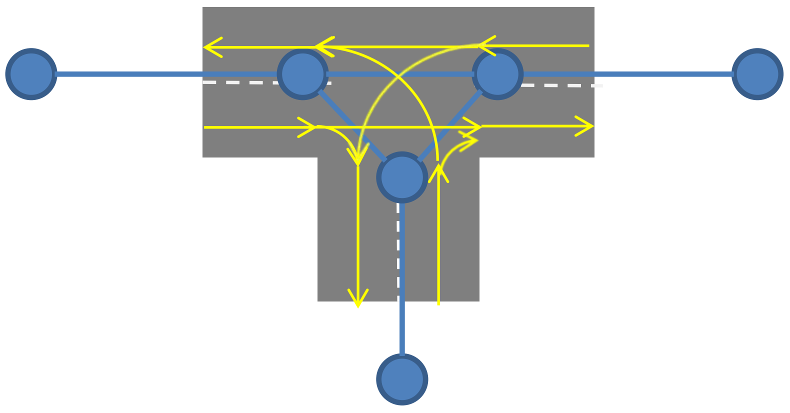


Figure 5 Possible ways to be chosen by the driver

Figure 5 shows the connectivity inside a T-intersection, which can be described with the new HD maps. HD maps has many more details or in other words less “blank areas”. By adding these enriched information of intersections internal connectivity, all possible ways a driver can choose, can now be described.

2.2 Reference model

A straightforward approach to make map data accessible for other applications is to transmit a part of the road network topology and geometry. The road network is a more or less densely connected graph resulting in a rather complex sequence of protocol messages to allow the unique identification of the links and nodes creating the road network graph. Since the vehicle is driving, the transmitted graph is constantly changing, i.e., links and nodes have to be added and removed.

The approach of ADASIS v2v2 and ADASIS v3 is to switch from a *map-centric* view to a *vehicle-centric* view of the world to create a simpler representation scheme:

A driver of a vehicle cannot make a choice at each intersection to drive in all possible directions at the same time traversing through the complete road network. Or in other words: a vehicle drives through the road network on exactly one unique route. This route is called a *path*. While the term road refers to the physical, real-world infrastructure constructed for personal mobility and transportation, a path represents one possible way along which the vehicle can move through the road network.

Using this assumption, it is not necessary to transmit a part of the complete road network. Instead, the road network can be reduced to a single linear map extract. Further, there is a much simpler way to make map data accessible: transmit only the map data along this route or path. (Figure 6 shows an example of a road network with the driven path.)

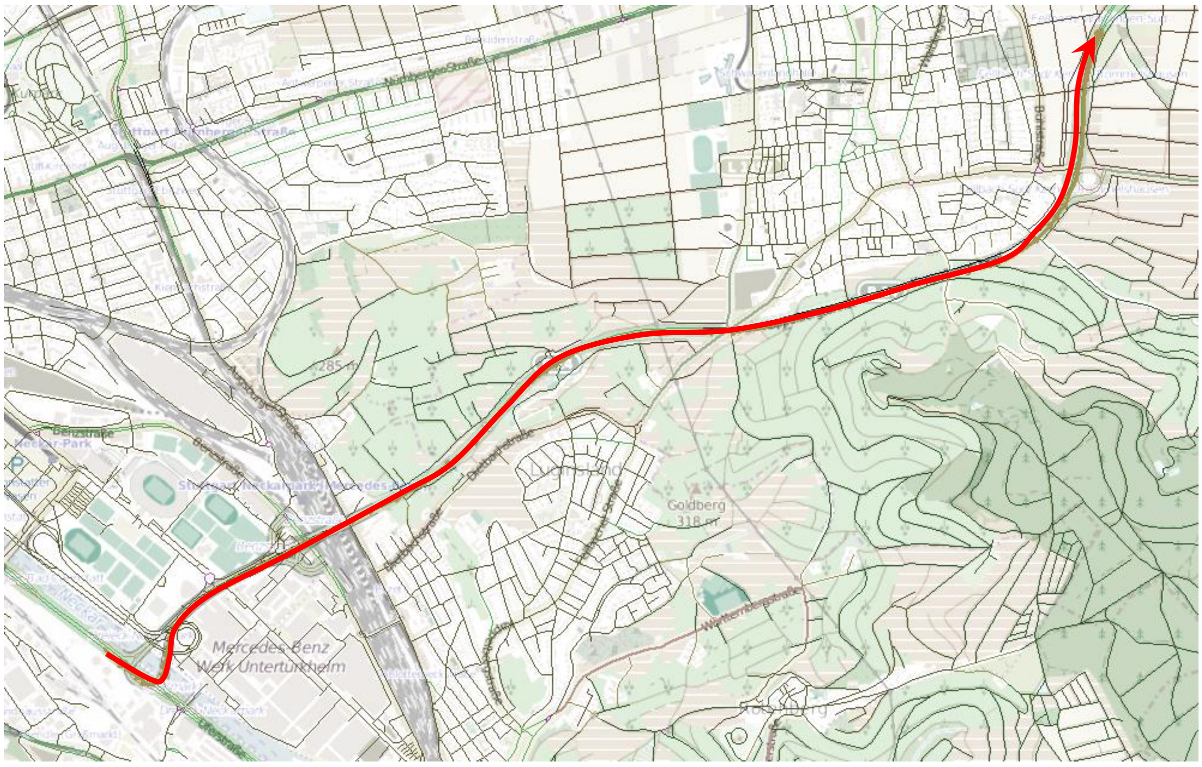


Figure 6 A digital map with the road network and a path showing a unique route driven by the vehicle

Using the abstract road network from the databases, such a driven path, or all possible paths, can be calculated with the current vehicle position as starting point. This is illustrated in Figure 7.

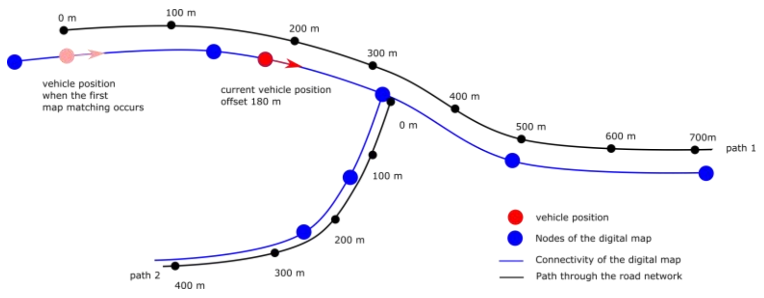

0 m
100 m
200 m
300 m
400 m
500 m
600 m
700m
path 1
vehicle position when the first map matching occurs
current vehicle position offset 180 m
0 m
100 m
200 m
300 m
400 m
path 2
vehicle position
Nodes of the digital map
Connectivity of the digital map
Path through the road network

Figure 7 - A simplified representation of vehicle paths based on the road network.

Paths are created using the road network of the digital map. This linear path consists of a sequence of street links and intersections. Using the path concept, various attributes from the map data can be transmitted indicating their location along this path. Figure 7 shows a simplified representation of vehicle paths that is abstracted from the physical road network (including reference nodes and branching points).

On a linear structure, the natural coordinate system to use is a distance marker from the origin of the path or a reasonable zero point. This distance marker is called the *offset* on a specific *path*. The point of origin, whose offset is 0m, is typically the start of a path at an intersection, the start of the driven route, the last intersection, or any reasonable point near the starting vehicle location (also see Figure 7: path 1 originates near vehicle location at first map matching time, path 2 starts at branching point).

While on the first glance, the current position of the vehicle would seem a logical point to measure offsets from, it is not used for this purpose in ADASIS. Most of the map data is assumed to be located at a fixed position. Thus, it is more worthwhile to keep the location of each map data attribute at a constant and fixed *offset* value. The vehicle itself is a moving object both within the coordinate system of the map data as well as within the newly introduced path-offset coordinate system. The location of the vehicle is represented with a changeable offset value. The vehicle offset value can be transmitted with a fixed rate (e.g., every second, every fraction of a second), or if the offset value changes when the vehicle moves, or using a mixture of both. Distances between two map data entities or between the vehicle and a map data entity is calculated as the difference between two offset values.

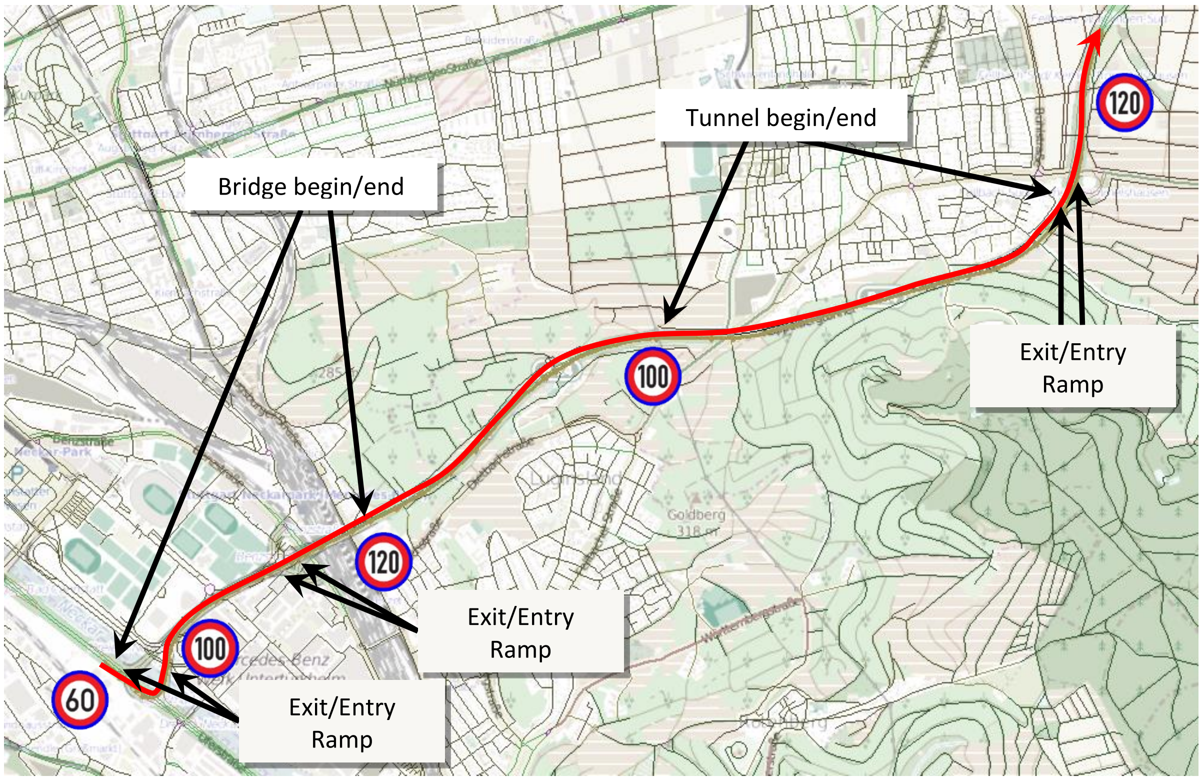


Figure 8 Exemplary map attributes along a route

Because we do not know the driven unique path, as the driver will not necessarily choose its way through the road network beforehand, we have to introduce means for representing deviations from the presumed unique path.

Deviations from a path can occur at each intersection or any branching point. Each deviation from the presumed path, our unique path, can be represented as an additional path the vehicle could drive through the road network. These additional paths can be extracted and transmitted from the road network in advance, too. The result is an extension of the single path representation to a multiple paths representation, for which the same representation scheme can be used.

A location in the linear path coordinate system is now uniquely represented by a couple consisting of a path identifier and an offset value specific to this path.

2.3 Path Representation of the ADAS Horizon

An abstracted and efficient representation of the road network is needed for transmission of the ADAS Horizon between provider and reconstructor. The horizon provider determines a set of possible vehicle paths through the road network including parent-child relationship information that is further described in the following subsections.

2.3.1 Representation of relevant links for ADAS Horizon

In the digital map database, the road network (Figure 9) is represented as a collection of *links* (or *segments*) and *nodes* that define connectivity between links (Figure 10).

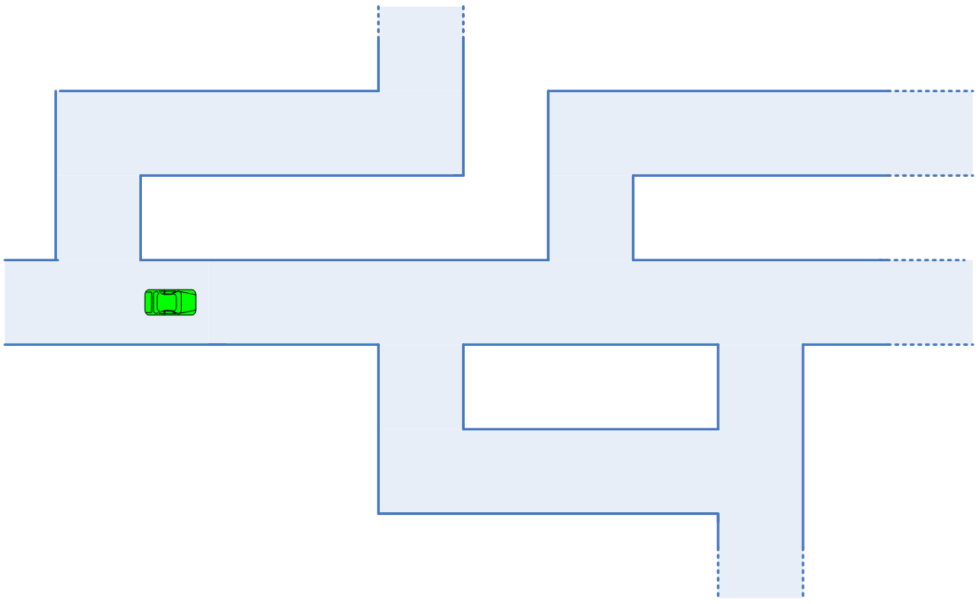


Figure 9 Road network

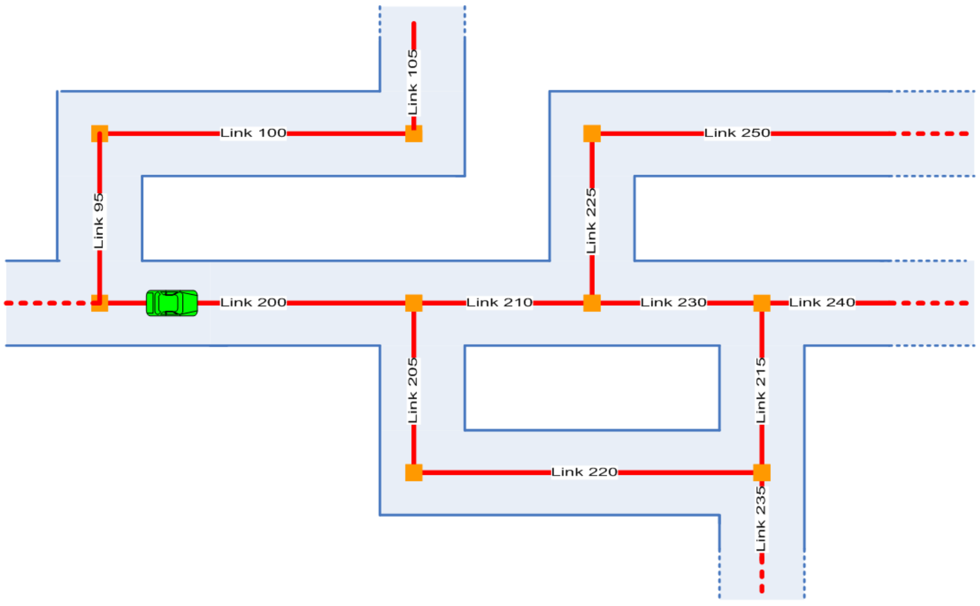


Figure 10 - an Exemplary network of links near vehicle location

With regard to the map-enabled and map-enhanced ADAS applications, the only links of interest are those that are ahead of the vehicle and accessible in a reasonable time.

The ADAS Horizon is the part of the digital map that contains only those road links in front of the vehicle (Figure 11).

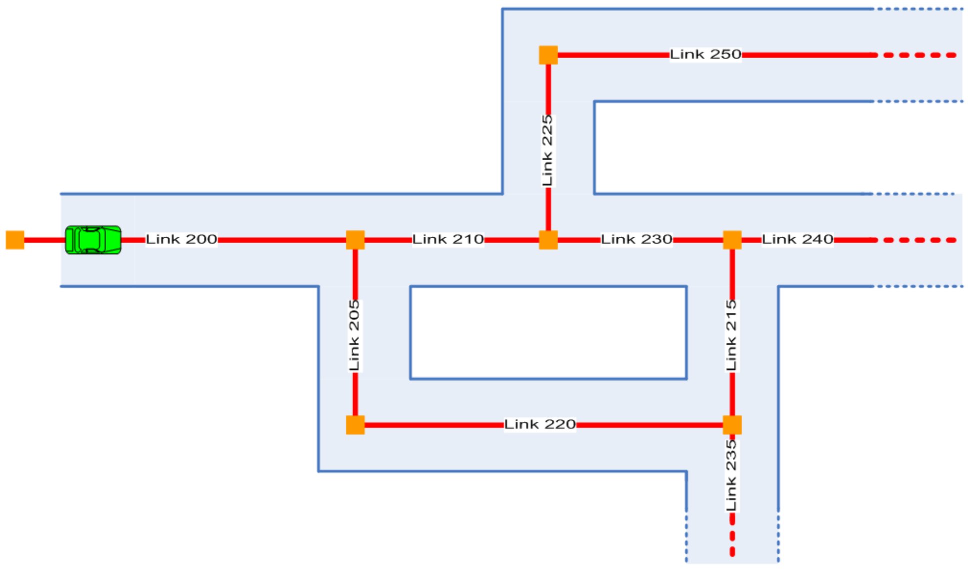


Figure 11 Relevant road links for ADAS Horizon

Comparing a simple extract of roads around the vehicle in Figure 10 and the relevant set of links for the ADAS Horizon in Figure 11, one can see that links 95, 100 and 105, being not important for the majority of ADAS applications, are not considered for the ADAS Horizon. In addition, not all link attributes available in the digital map need to be present or used for ADAS Horizon creation. For instance, street names or house number ranges will rarely be needed by ADAS applications.

In other words, the ADAS Horizon provides to ADAS applications an optimized view, path-based view of the environment, which is described in the following subsection and allows for more efficient processing.

There are different algorithms for the construction of the ADAS Horizon. However, descriptions of those algorithms are not within the scope of this document.

2.3.2 Simple Path representation of ADAS Horizon

The ADAS Horizon can be created and abstracted from a *Network of (Digital Map) Links*, as shown in Figure 11. For the ADAS application, this view is rather complex because it must analyze the whole network of road links in order to get information about link characteristics and map attributes in front of the vehicle. For instance, if there is a traffic sign of interest on link 235, the application must realize by itself that this link can be reached by following links 200→210→230→215→235 or 200→205→220→235.

Instead, it is more convenient and less complex for the ADAS application to deal with *paths* – possible routes the vehicle may follow in the near future. Inside the Horizon Provider, paths are built from map database links and their connectivity, but each path is seen by the application as a single entity.

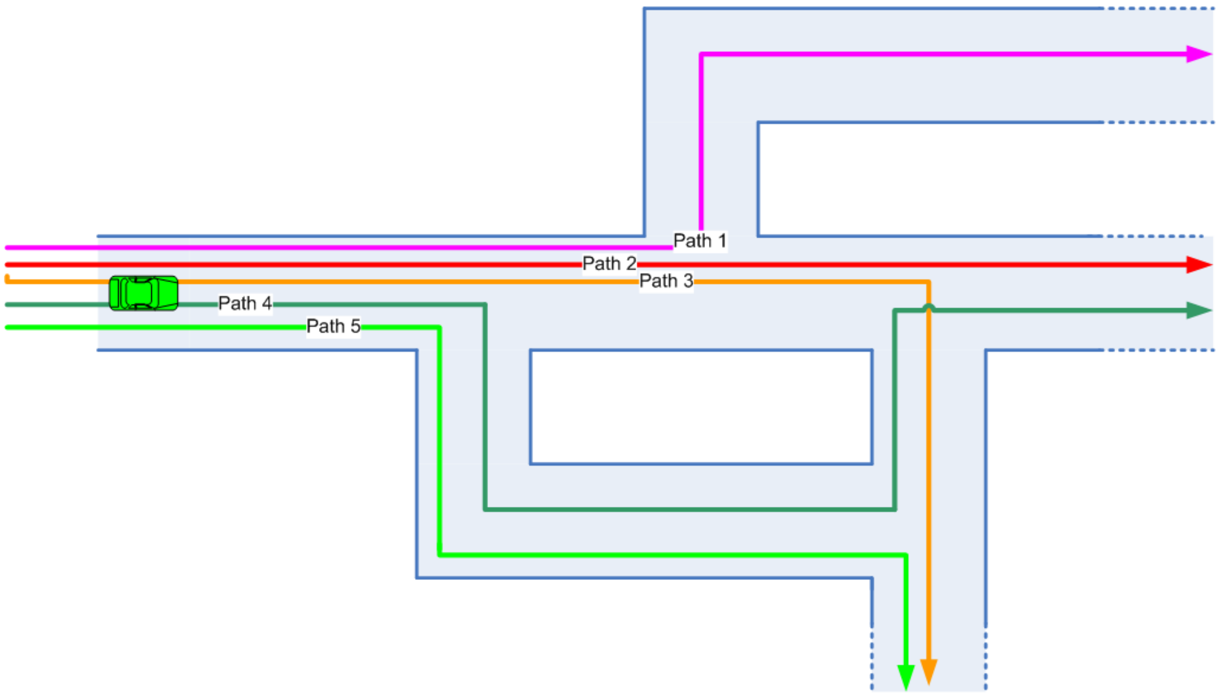


Figure 12 ADAS Horizon – Pure Path representation

Coming back to the original example, by querying path characteristics, the application can now easily recognize that there is a specific traffic sign on paths 3 and 5 (Figure 12). It may or may not be known that traffic signs on those two paths are the same physical traffic sign. For most applications, however, this information is not fundamentally important – the only significant information is that there is a traffic sign ahead and the distance to it.

Pure Path Representation, as shown in Figure 12, contains a number of redundancies. For instance, all five paths have a common start point. This characteristic is not very desirable if the ADAS Horizon is to be transferred over a slow communication channel, since transmission of redundant information consumes more bandwidth than actually needed.

Optimized Path Representation in ADASIS v3 reduces, but does not fully remove, such redundancies.

2.3.3 ADASIS v3 Optimized Path Representation of the ADAS Horizon

The ADASIS v3 concept describes an ADAS Horizon using *Optimized Path Representation.* This approach reduces the amount of redundant data, but still provides most of the advantages of the path approach over a road network representation of the ADAS Horizon (Figure 13).

As a minimum for using Optimized Path Representation just one path needs to be present, which is referred to as the *root path*.

The vehicle may turn from the root path to *Sub Paths*. In Figure 13 Paths 1, 3 and 4 are first-level sub-paths of the root path.

At first-level sub-paths there are possible turns to higher level Sub Paths (Path 5, for instance, from Path 4), etc.

The ADAS Horizon's construction algorithm should choose the root path so that it appears to be the most likely alternative for the vehicle to continue driving. First-level sub-paths are less likely to be driven on and so on.

Most ADAS applications operate only on a single path. In the simple case this is the path that the vehicle is currently driving on (the *vehicle path*). But information learned after creation of this path might make it likely for the vehicle to turn onto a sub path in the near future. In this case applications would prefer to use a path starting at the current vehicle position and continuing on that sub path. This is what is called the *Most Preferred Path*, which is not a path in the meaning of a fixed branch of a horizon tree, but a combination of parts of paths which make up the way through the horizon tree that the vehicle currently is expected to follow.

In distributed environments, with one ADASIS v3 Horizon Provider and one or more ADASIS v3 applications, the decision on how many levels of sub-paths should be sent depends on the application's tolerance to lack horizon data. If few seconds of blindness are acceptable when leaving the MPP and switching to a new MPP, only a single path data would be sent over the communication channel. After vehicle has left this path, the ADAS horizon provider needs to establish and send a new one, which will be accomplished within a short time span.

If an application requires having complete horizon data all the time, the full ADAS horizon (including all possible paths) must be transferred. This ensures that, even if vehicle leaves one path, it will automatically be positioned on another path that is already available on the client side.

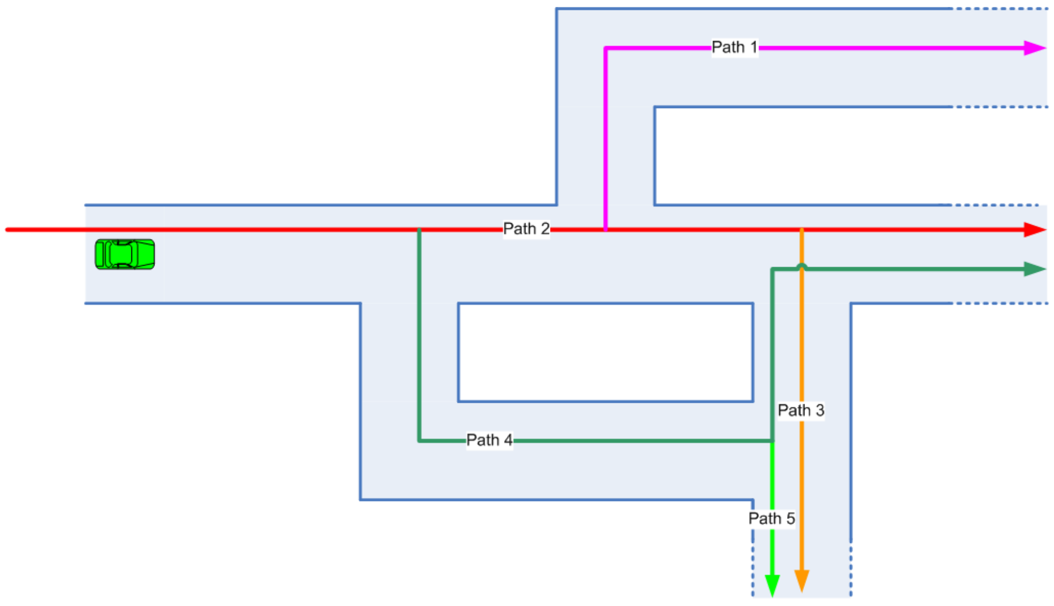


Figure 13 ADASIS v3 Horizon - Optimized Path Representation

Since ADASIS v3 implements the ADAS Horizon using Optimized Path Representation, let us call this representation the *ADASIS v3 Horizon*.

On the ADAS Horizon, crossings, road attributes and even geometry may be seen only as characteristics of (one of) the paths. Therefore, the *path* is the main entity that needs to be accessed by the application in order to retrieve the desired information.

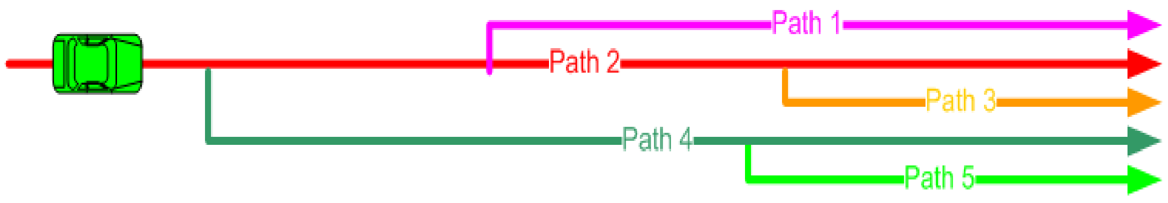


Figure 14 : ADASIS v3 Horizon - Application View

From the application viewpoint (Figure 14), the vehicle is located on one path or on a second path as alternative position. The data of interest is either on the same path or on one of sub-paths that are part of the ADAS horizon.

2.4 Branching depth of ADASIS v3 Horizon

ADASIS v3 Horizon branching depth can be limited to different levels, with each level adding more information on the Horizon tree to the previous level. The levels of branching are:

- **Single path only**: without side paths, detailed information relating to other paths cannot be provided.
- **Drivable paths**: only paths that our own vehicle might use will be included.
- **All paths**: additional paths are included that describe possible trajectories for other road users. This level can be augmented with collision points.

Additionally, the provider will have an upper limit for the total number of paths, further limiting the branching of the Horizon.

The Horizon will only include information for the paths that are present in it. Therefore, while a single path representation is able to indicate branching nodes for child paths, the details of those child paths will not be available to a reconstructor.

3 ADASIS v3 Basic Entities

As we have seen, the ADASIS v3 Horizon consists of *paths*. Each path is uniquely identified by a *path identifier.*

Paths can branch off from a *parent path*, making them sub *paths* of their parents, or they can be *root paths* that do not have a parent. The start points of side paths are defined by the parent path and by the *offset* from the start of the parent path.

Properties of paths are described by *profiles* consisting of *profile entries*; these are also located on the ADASIS v3 horizon by the path they apply to and by their offset from the start of this path.

The current vehicle status is described in terms of its *position* with respect to the ADASIS v3 Horizon (again path identifier and offset) and other characteristics such as speed and heading.

3.1 Path

The ADAS Horizon generally is a *predictive tree view* of the surroundings in front of the vehicle and it is described as a collection of different paths and digital navigation map data along them. The predictive tree is built up of connection of several paths, each one representing a part of the road and connected through *intersection points (*crossings) to the others. As soon as the vehicle moves and changes its position, the predictive view changes too: some passed paths might be removed and/or new ones might be added.

A path is characterized by a collection of attributes (data describing the surrounding environment and the street itself, as the number of lanes, geometry, curvature, etc.), *offset values* and *parental relationship*.

Offset values are a kind of distance marker and locate virtually attributes/data along a path by defining the absolute distance, expressed in centimeters, along the path itself. A path always begins with an offset value of zero associated with its origin point and the offset value of its attributes indicates the distance between the attribute itself and path's origin. This value increases for the whole path's lifetime. If a path is started anew and there is no parent path, the offset value 0 is located at a reasonable point near the first vehicle position match.

At the implementation process, the system engineer shall define a maximum *Horizon Length;* within a specific view/run, the provider shall not exceed this value for the managed data. Horizon length is defined as the distance between the smallest managed offset in the horizon tree to the maximum offset in the horizon tree:

horizon_length = greatest offset – smallest offset.

The provider can build up a path either with a smaller path length or up to the maximum defined value.

No gaps shall be introduced while building up a ***specific Horizon tree***, i.e. each path shall be connected to another one (except for a *root path*). Each path must have only one parent path, from which it branches, and may have zero, one, or several child paths (sub paths). The new path starts at the split point of the lanes in the parent path if available from the HD maps, or at the intersection node otherwise.

3.1.1 Definitions

ADASIS v3 introduces some path definitions/concepts, some of them already available within ADASIS v2, other completely new or different.

- Root Path;
- Vehicle Path;
- Most Preferred Path (MPP).

A *Root Path* is a path originating a Horizon tree; therefore it has no parent path. Within a specific horizon tree only one root path shall exist at once. All other paths shall be connected to each other and reachable from the root path itself.

The *vehicle path* represents the path which has the highest probability for the car to be on, according to the probabilities of the different map matched positions.

The *Most Preferred Path* (MPP) represents the path with the highest probability that the vehicle will follow this route. It always starts from the path on which current vehicle position is on and ends up to the path defined as preferred path within the position message. The complete chain between this end path and the current vehicle path can be derived by a path to parent path relationship, defining the route from the current vehicle path up to the expected destination path, on which it is expected that the vehicle will arrive. This concept substitutes the ADASIS v2 Most Probable Path and could be considered as a kind of "overlay" over the 2d path model.

3.1.2 Alternative trees

In ADASIS v3, it is not required that all paths form one single tree originating from one root path. Rather, there can be multiple trees, each with its own root path. So the overall ADASIS v3 horizon can have the structure of a forest.

Such a forest can be the result of having multiple possible vehicle positions on roads not connected to each other. In this case, there is a main vehicle position, given first in the list of possible positions, which defines a main tree. (Usually this is the vehicle position with the highest probability, but this is not required by ADASIS v3.) Other (alternative) vehicle positions may lie on the main tree as well, or they may lie on additional unconnected trees, called alternative trees.

Furthermore, there can even be paths and trees not directly related to a vehicle position (called non-drivable trees).

3.1.3 Horizon Topology

To form the horizon, the paths are connected to each other in a parent-child relationship in the driving direction.

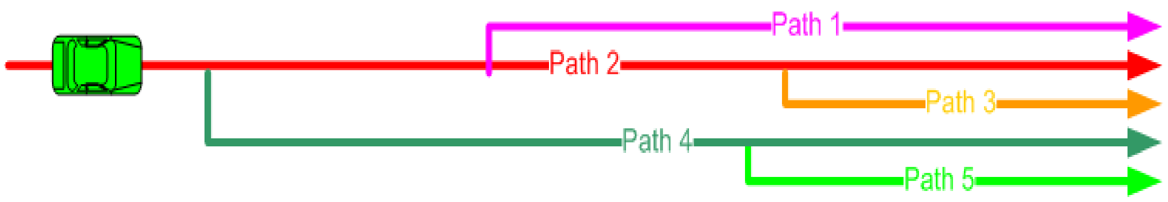


Figure 15 ADASIS v3 Horizon Topology

In Figure 15, Path 1 is a child of its parent Path 2; Path 5 is a child of Path 4 which is a child of Path 2.

3.1.4 Horizon View

This definition is view/context dependent:

- **Static View**: The system always uses the path the vehicle currently is driving on as the Most Preferred Path (MPP). At an intersection, a path leading up to the intersection is extended with the arm with the highest probability, or with the arm corresponding to a route from a navigation system if available; other arms of the intersection can start side paths. The MPP will not be changed by the arrival of new information that would influence probabilities, or by a new route from the navigation system.

- **Dynamic view** (route/destination can change): the MPP can change (resulting in an MPP different from the current vehicle path)
  - if the navigation system changes the guided route (to reach the destination),),
  - because of dynamic probabilities,
  - if the user selects a new destination.

3.2 Profile

Profiles provide attribute information along paths in the Horizon in a generic way. Each profile has a type, which defines the kind of data stored as the values of the profile.

For example, a speed limit profile provides speed limit values for a point along a path.

3.2.1 Definition

A path can have any number of *profiles*, each uniquely identified by a profile type. A profile describes the value of some attribute of the road network along a path. Individual *profile entries* describe the value of that attribute at a given location (i.e. offset) on that path. The profile entries for one profile type (and one driving direction, if the profile is direction dependent) together constitute a profile.

It is allowed in an ADASIS v3 environment to send profile entries of some type only for locations where they have values of interest and to leave out profile entries for locations where e.g. a default value applies (e.g. one could send a “tunnel” profile entry just for tunnels and send nothing outside of tunnels); if doing so, though, it is not easily possible to recognize on the receiver side whether data has been received completely.
On the other hand, for some profile types (e.g. traffic signs) it is allowed and reasonable to have multiple profile entries at the same location, i.e. at the same offset on the same path. And some profile types allow lane specific entries, in this case again multiple profile entries of the same type can occur at the same location.

The value stored in a profile entry is named profile value. The profile value has a specific data type which is defined by the profile type.

The location of a profile entry along a path is defined by an offset on the path. The offset is defined by its distance from the path origin, as described in 2.2. The values of a profile in between these stored locations are determined based on the interpolation type of profile. Depending on the interpolation type, a profile value may also be defined as a vector containing several values necessary for smooth curve representation (e.g. for polynomial or spline interpolations or linear interpolation with unsteadiness).

The following rules apply to a profile of a specific profile type:

- Whenever possible, only changes in attribute values shall be transmitted in a profile entry, except for profiles with spot interpolation. See below.

- Data may be missing for profile values at some locations of a path. In this case the profile entry represents the non-availability of a value using the flag *NotAvailable.*

Some profiles depend on driving direction (e.g. speed limit or number of lanes). In these cases it is possible to provide data separately for each direction – in path direction, and against path direction. The profile entries of the same type for different direction are handled separately (e.g. for completeness checking, updates, etc.).

Note that this is relative to the direction of the path, so the two possible directions are “along path direction” and “against path direction”. For a path that could be reached by the ego vehicle, this is equivalent to “in my direction” and “against my direction”, but the definition also applies to unreachable paths. In particular, if a sub path represents a one-way road leading up to an intersection with its parent path, all vehicles will drive “against path direction”, as the sub path is starting at and pointing away from the intersection; profiles “along path direction” in this case will not be particularly useful.

Offsets for profile entries valid against path direction are still measured in path direction, i.e. the endOffset of a profile entry is greater than the offset (the offset range just describes a portion of the path).

3.2.2 Interpolation types

Interpolation should be performed to calculate profile values for intermediate points in between two profile values, depending on the profile type. It describes a profile as a function of an offset along the path. The interpolation type is defined for a profile type and therefore is fixed for one profile.

The following interpolation types are supported by ADASIS v3.

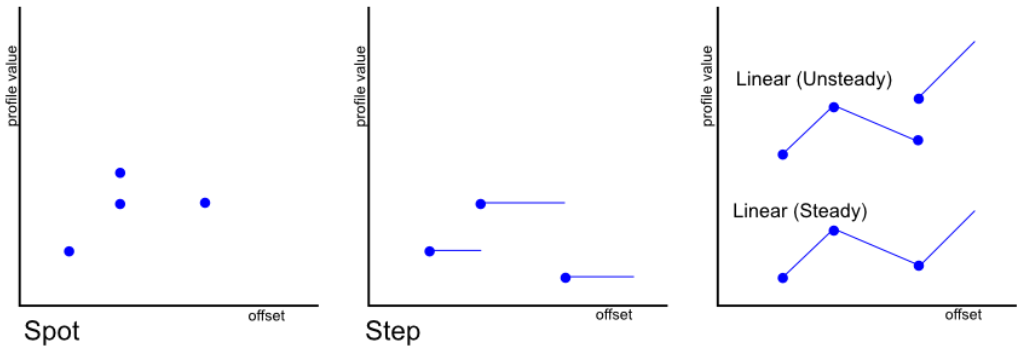


Figure 16 Different Interpolation types

1. Spot interpolation

Profile values in a profile with spot interpolation are only valid at the given offset of the profile value. The result of reading intermediate values is an implicit non existing value. Consecutive spot profile values can be identical. In this case the attribute change is expressed by the offset.

For example, a traffic light can be defined with a spot profile because the traffic light is located at a specific offset on a path and intermediate values are not meaningful.

Multiple spot profile values can be stored at the same offset for specific profile types, e.g. traffic signs.

2. *Step interpolation*

Profile values defined with step interpolation are valid until the offset of the next profile value. The profile value at an intermediate offset is the profile value before (or at) the offset.

For example, a general speed limit is constant for a certain distance until the next change.

A step profile can hold multiple values at the same position but overlapping is not allowed. With every change in the profile, all values must be repeated. As a result, at any offset with a change in one or more profile values, all existing values at this offset are given.

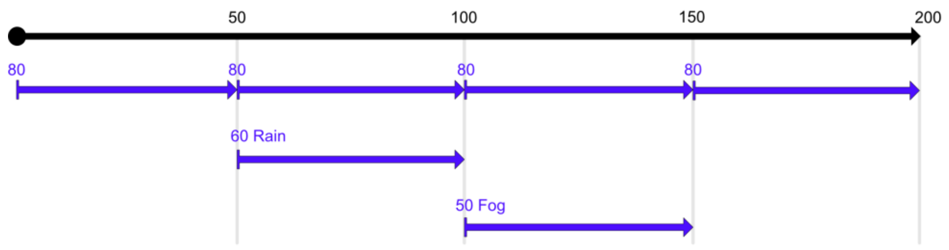


Figure 17 Example for multiple profile values at the same offset

The example in Figure 17 shows a path of length 200. A simple speed limit of 80 is valid for the whole path. Additionally, there are 2 restricted speed limits starting at offset 50 and 100. The profile values are:

- Offset 0: speed limit value of 80 is introduced;
- Offset 50: new speed limit of 60 in case of rain is introduced, speed limit 80 continues and is repeated;
- Offset 100: speed limit 80 is repeated because the limit 50 in case of fog starts and the limit 60 in case of rain ends;
  Offset 150: speed limit 80 is repeated because speed limit 50 at fog ends.

3. *Linear interpolation*

Profile values for a linear interpolation profile (first order) are specified at certain locations, and the values must be linearly interpolated between the given locations. This approach is appropriate if the values of the profile are continuous, i.e. change steadily.

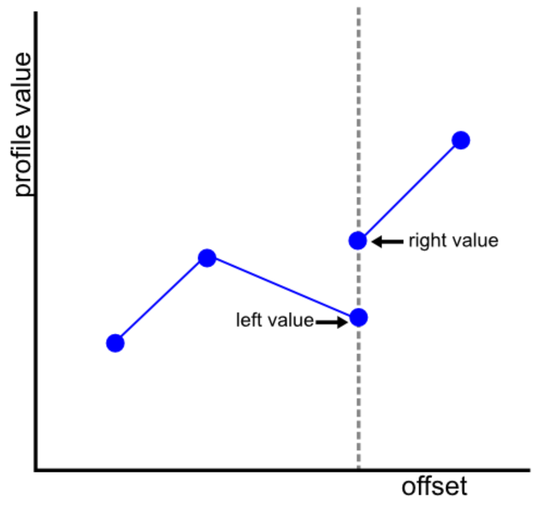


Figure 18 Example of a profile with discontinuous linear interpolation

A linear interpolated profile is discontinuous when, at some offset, the left value (interpolation end value when approaching the offset from the left) is different from the right value (interpolation start value when approaching the offset from the right). To model a discontinuity in the values, the following approach is used. The left value is stored in the profile entry with endOffset=0, and the following right value is stored at the same offset, but with endOffset>offset (see Figure 18).

Discontinuity is the only case where two values appear at the same offset in linear profiles.

4. Higher order interpolation profile

Higher order interpolation profile (quadratic / cubic polynomial, spline, etc.) are specified by a set of locations with defined sets of attribute values at the locations. The interpolation algorithm to be used to calculate the profile value between the given locations is specified by the profile.

5. Special interpolation

For slope and curvature, multiple pairs of offsets and values are packed into a single profile entry in order to save bandwidth. Slopes and curvatures can be interpolated, but this cannot be done in the standard fashion as the value of a profile entry is not a scalar value but a packed data structure. For interpolating, a client needs to understand the details of these data structures and to handle interpolation in a way depending on the specific profile.
The “special interpolation” just describes the circumstance that interpolation is possible, but only with special knowledge. Other profiles might be added in the future having this interpolation type and their own rules for handling interpolation.

3.2.3 Profiles Hierarchy

The introduction of HD maps, which extend the number and size of map data on the roads, has led to the drastic increase in amount of transmitted data. As a result of this increase, the bandwidth between the provider and the reconstructor is required to be high. Profile hierarchy is introduced to decrease this bandwidth requirement. This hierarchy allows the reconstructor to choose which groups of profiles it needs, which results in the optimization of bandwidth between the provider and the reconstructor.

Profile hierarchy is done on the logical representation of the ADAS Horizon. The first and also basic level is the profiles which belong to the path. The next level is made of the profiles which describe, in more detail, each lane in the road. And the last level is made of the geometry profiles.

In case of the reconstructor only needs the basic path data, the provider can avoid sending data from the other levels to decrease the amount of data which has to be transferred. It is up to the developer of such profiles to define which data should be located in which level.

3.2.4 Global Data

Global data is a generic version of the metadata message in ADASIS v2. It allows the transmission of horizon related data which is not attached to a specific path. Global data is sent using the GlobalDataMessage ([2]).

There are some use cases for Global data which are:

- to provide data for the current position (e.g. country and region code or the current speed unit);
- to provide status information (e.g. version information or guidance status);
- to provide events (e.g. a change to the current distance to guided destination or global weather updates).

To simplify the concept, global data and horizon path data are using the same profile data structure definitions. Profile types can be used as global data, as path data, or both. One GlobalDataMessage can hold a variable number of profile entries. Global data is usually sent on change or within a regular interval. Global data can be split among different GlobalDataMessage instances. This allows for different timings of the contained data.

In general, the GlobalDataMessage allows sending arbitrary data as events. However, this is not recommended, global data should be related to map data, provider status, or ADASIS protocol version.

3.3 Position

The set of paths represent parts of the street network as predictive trees using our constant path-offset coordinate system. The *Position Message* is used to represent the position of our ego vehicle on zero, one, or more paths of these predictive trees. The position information is typically regularly transmitted to describe a moving vehicle, and to inform the receiving application of the current vehicle position status.

The vehicle position on a path is described as a coordinate (*path*, *offset*). Often the vehicle can be positioned on more than one path with some uncertainties. This leads to the generic position description using an array of position coordinates with an added position *probability* and, as an additional hint, a *deviation* distance between the calculated absolute vehicle position and its map-matched position on a path in an implementation specific way. The array can be empty to represent the situation where the ego vehicle cannot be matched. The following states can be represented:

- Vehicle is off-road,
- Vehicle is on-road, i.e. map-matched on a path,
- Vehicle is on-road on more than one path,
- Vehicle is off-road and near a path with a map-matched position on this path,
- Vehicle is approaching or leaving a path.

Additional information such as the existence of map data, vehicle absolute position information, other calculations and system state are not transmitted in Position messages. Such information can be transmitted using Global Data messages.

A *timestamp* is explicitly added to the vehicle position information. This timestamp represents the time of the collected sensor data values used in the calculation of the vehicle position. This timestamp does not represent the time when the path/offset coordinate was calculated, an absolute vehicle position is calculated, or the Position message is transmitted. It represents the time of the used input sensor data, e.g., the time when the GNSS absolute vehicle position was received or the time of the sensor data for dead reckoning. The timestamp is given in a global time shared by all ADASIS Horizon providers and receivers. If no shared global time exists, a *position age* or time difference between this timestamp and the time when the position message is transmitted on the used transmission medium shall be calculated.

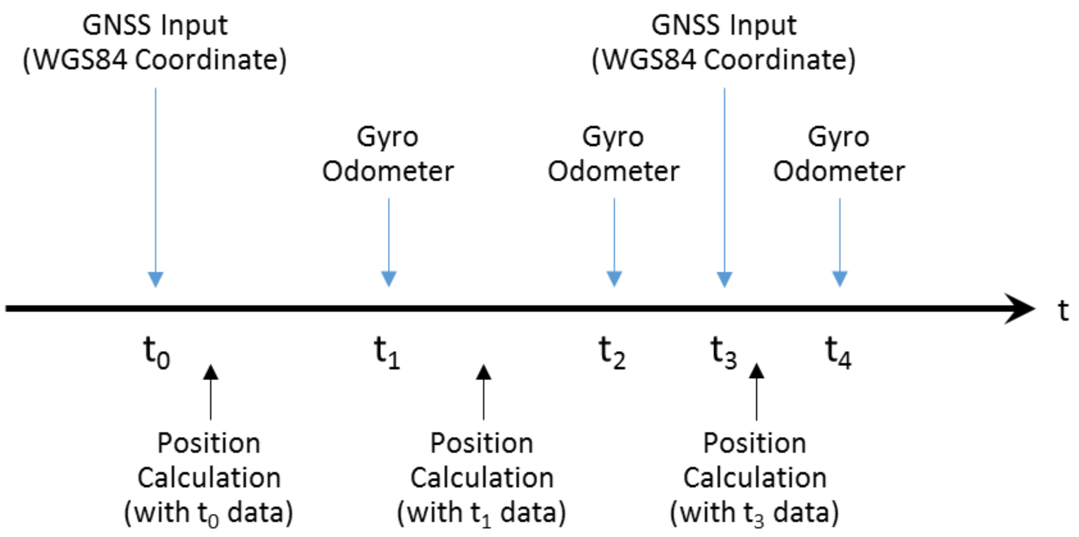

GNSS Input (WGS84 Coordinate)
Gyro Odometer
Gyro Odometer
GNSS Input (WGS84 Coordinate)
Gyro Odometer
t
t_0
t_1
t_2
t_3
t_4
Position Calculation (with t_0 data)
Position Calculation (with t_1 data)
Position Calculation (with t_3 data)

Figure 19 Timestamp used for a vehicle Position information

A *deviation* is an explicit part of the vehicle position information. This value represents a distance between the absolute vehicle position and its projection on the map representation of the road, road lane, or road surface. If the vehicle can be map-matched on one or more paths with deviations, the deviation value can be used as an additional indication of the quality and uniqueness of an on-road position. If the vehicle is in off-road state and is approaching or leaving a road link, the deviation value can be used in an application as supporting decision criteria to use or not use the horizon data. The exact definition of the deviation is implementation-specific, see [2].

A *preferred path* is an explicit part of the vehicle position information. This value can be used to describe a most preferred path (MPP) starting at the given current path, but which does not necessarily follow this single path but can branch off to sub paths of the current path. The referenced path given by the most probable entry in the Position message array represents the MPP. Usually the ADASIS Horizon provider assumes that the driver will follow this path and the provider will build and extend this path. However, additional information from other systems (e.g., route guidance) can trigger a change of this MPP onto a sub path in the horizon tree in front of the vehicle. This MPP can simply be indicated and updated by tagging one of the tree path as *preferred path*, in addition to the current path the position is on. It is indeed sufficient to reference only the current path and the preferred path to uniquely describe a most preferred path as a sequence of paths using the parental information of the paths.

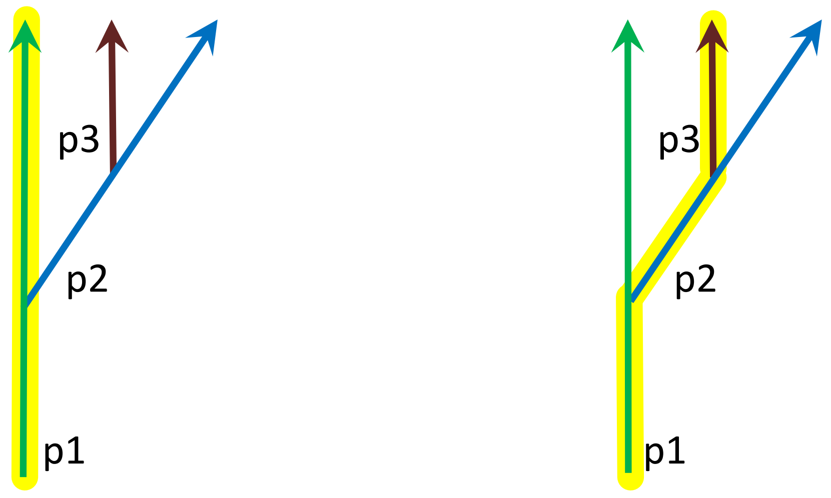


Figure 20 Left: current path and preferred path are both p1. Right: current path is p1 while preferred path is p3. The parent-child relationship from p3 to p1 describes the MPP

The complete vehicle position information consists of a timestamp and an array of position entries. Several situations may occur:

- There are no entries, i.e., the array is empty, if no absolute vehicle position is available or no map data for the vehicle position is available or any other system-dependent reason that information is missing.
- There is exactly one entry with state *off-road* (Path Identifier == 0), if the absolute vehicle position is known and map data exists, but the vehicle cannot be map-matched on a path.

- There is one entry with state *off-road* (Path Identifier == 0) and there are one or more entries with state *on-road* (Path Identifier > 0), if the system can map-match the absolute vehicle position to one or more paths, but the system-dependent criteria for a successful map-matching is not met. In this state, the deviation values of the entries representing the on-road positions shall indicate the distances to the map-matched path candidates.
- There is one entry with a state *on-road* (Path Identifier > 0), if the system can successfully map match the absolute vehicle position to one path and if there is no other on-road candidates on other paths.
- There are two or more entries with a state *on-road* (Path Identifier > 0), if the system can map-match the absolute vehicle position to more than a path. In this state, the first entry is the most preferred position, i.e. the position that the ADASIS horizon provider recommends clients to use. This does not always have to be the position with highest probability, e.g. if there are two positions with almost equal probability, then in each map matching cycle another one might get assigned highest probability, but the provider still can ask clients to stick to one of the positions by listing it first, even if for a time it only has second highest probability. The other entries are additional position candidates and can be used to get additional information about parallel or crossing roads. The probability, accuracy and deviation values in each position entry can be used as an indication of the quality and uniqueness of the candidate.

4 ADASIS v3 Management

The basic structure of an ADASIS v3 system consists of a horizon provider supplying ADASIS v3 Horizon data to one or more reconstructor applications, as illustrated in Figure 21.

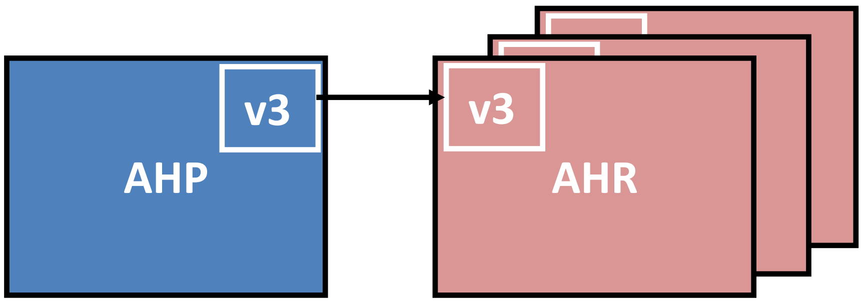


Figure 21 Basic structure of ADASIS v3 System

These applications might be connected to the horizon provider via some in-vehicle network, or they might run on the same computer system as the horizon provider and be connected via some local communication mechanism.

4.1 ADAS v3 horizon synchronization between provider and receivers

In ADASIS v3, the provider is responsible for managing the active contents of the horizon. Two different concepts are mandatory to exchange this information between the provider and the reconstructor, which are *Path Control* and *Profile Control*. Path Control manages the topology of the horizon, and Profile Control manages the contents of the horizon paths. The two concepts are described in more details in 4.1.1 and 4.1.2.

The advantages of the new mechanism will be:

- Unique identification of what information (data and path) becomes obsolete.
- Dynamic behavior during lifetime.
- Only one implementation (provider side) in the complete vehicle is needed.
- Separation between path lifetime and profiles lifetime.

The provider gets full control over the horizon content. It not just transmits new information to a receiver – the provider also decides when a receiver should delete data from its local storage. This way the provider can know what data receivers have in memory, and thus how much additional data can be stored. As a result, the provider knows whether it can send some data item without overflowing memory on the receiver side.

Rules for paths lifetime are known only to the provider and are fully transparent to receivers.

4.1.1 Path Synchronization

Paths are synchronized between provider and receivers using dedicated Path Control Messages which defines the full list of paths that should exist in the ADASIS horizon.

Paths listed in a Path Control Message are created locally by a receiver if they do not already exist there (i.e. the receiver creates the internal data structures that it needs to manage a path). Paths that exist locally in a receiver but are missing in a Path Control Message are deleted locally by the receiver.

It is expected that a provider sends Path Control Messages at regular intervals to make sure that receivers stay synchronized (or re-synchronize) in the case of an occasional dropped message. A provider might send additional Path Control Messages to make a new path known to receivers or to instruct receivers to delete paths that are not needed anymore in order to make room for new data. (The vehicle environment might limit or restrict the sending of additional Path Control Messages beyond the regular cyclic ones, though.)

In addition to the mere list of existing paths, the Path Control Message contains information about parent-child relationships between paths and about the offset on a parent path where a child path branches off. With this information, Path Control Messages define the complete topology of the horizon tree (or forest): after processing Path Control Messages a receiver will have sufficient information to store other information (profile data, in particular) and to attach it to the appropriate location in the horizon tree. This helps in fast startup of a receiver when the provider already has been running for some time, and in fast recovery after an error.

Please note that empty Path Control Messages request receivers to delete all horizon content (except global data); this is explicitly allowed and used to reset the ADASIS v3 horizon.

1. *Provider rules*

Whenever the provider adds a path to the ADASIS horizon, it assigns a new path ID to it; it will include this path in future Path Control Messages.

If the path has a parent, the ID of the parent path is included with the new path (0 is sent otherwise, i.e. for a root path). A provider should only use a parent path ID that refers to a path that exists and is known to receivers. The only change that is allowed for the parent path ID of any path is from non-0 to 0, this can happen when the parent path is removed. It is also acceptable, though, for the provider to continue sending the ID of the deleted parent path.

It is recommended that the provider either wait until a Path Control Message with the new path has been transmitted before sending data (profile data, in particular) pertaining to the new path, or that the provider sends an additional Path Control Message including the new path before sending data pertaining to the new path.

It is allowed, though, to send data, including profile data, pertaining to the new path before a Path Control Message with the new path has been sent. In the case of profile data, though, this will cause a receiver to locally create a path even without having a Path Control Message; this is called implicit path creation (while the standard behavior of creating a path listed in a Path Control Message is called explicit path creation). The provider has to consider that a receiver needs to store an implicitly created path and must not trigger implicit path creation if receivers would not have sufficient storage space available for the new path. The path starts at the path split point which is defined as where the lanes start splitting (in case provided by HD maps) or at the intersection point (otherwise).

It is up to the provider to decide when a path is no longer relevant and may be deleted. No rules are explicitly specified by ADASIS v3. A typical implementation would e.g. delete a path when every possible current vehicle position has moved away from the path for at least some configured distance.
Please note that this can lead to a path not being deleted for a very long time – as long as the vehicle is following the expected path (MPP). This is expected behavior, and ADASIS v3 is designed in such a way that this does not cause problems.

2. *Receiver rules*

A receiver will locally create a path listed in a Path Control Message if it does not yet exist locally (except if storage space is lacking for the new path; this would indicate an error, and then the new path will not be created locally).

A receiver will locally delete a path missing in a Path Control Message that still does exist locally. All profile data attached to the deleted path will be deleted as well.

If a receiver receives profile data (including a Profile Control message, see below) for a path that does not exist locally in the receiver, then the receiver also will create this path locally (except if storage space is lacking for the new path; this would indicate an error, and then the new path will not be created locally).

If a receiver receives other data containing the ID of a path that does not exist locally in the receiver, then the receiver will store the data as is (as a “dangling reference”). When using any path ID that could have been received in such a way, then the receiver will have to check the existence of the path referred by the path ID. This affects Position Messages, the parent path IDs in Path Control Messages, the sub-path IDs in Node profiles (see below), and the path IDs in Lane Connectivity profiles (see below).

4.1.2 Profile Synchronization

Profile Synchronization between the provider and reconstructor indicates which profiles are managed on each path of the horizon. Managing a profile means that the profile is stored on the provider side. Profile Synchronization is done for a specific range of offsets. This Range is presented by *Profile Control offsets.*

Profile Control offset is defined as the smallest offset on the path where profile entries are stored and handled on the provider side. All profile entries with a bigger offset than this *Profile Control Offset* are inside this range. Additional profile entries included into this range are those that partially fall into it. It happens when the start offset is before the range, while the end offset is still in.

All profile entries inside those ranges will be managed by the provider and could be updated every time. Therefore, all profile entries which belong to that ranges have to be stored and handled inside the reconstructor.

Rule: The *Profile Control Offset* of each path is transmitted via the *Profile Control Message*.

The provider sends a *Profile Control Message* to update the profile control offsets.

If the provider increases the start offset of a path under its control, it must send the new start offset to the reconstructor via *Profile Control Message* and then the reconstructor is free to delete the attributes before this offset. Paths unreferenced by a *Profile Control Message* must be kept unchanged.

Detection when path-data can be deleted

Path data deletion is handled through the *Profile Control Message*. This message is used to update the *Profile Control Offset* of some of the managed paths. The provider has to send this information for any path whose start offset has increased since the sending of the previous *Profile Control Message*. After receiving such message, all reconstructors can delete all path data related to an offset anterior to the updated value.

The reconstructor can keep data on a specific path if needed but it has to always keep profile entries with a validity range ending behind the Profile Control offset. In case the reconstructor decides to keep more data, it is solely responsible for managing this data.

4.1.3 Separation between path life time and path-data life time

In ADASIS v2 there was only the possibility to remove data by checking the system-wide trailing length. This check was done for data on the paths and also for paths itself by each receiver. When an intersection point was behind that trailing length, the parent path with all its data has to be deleted.

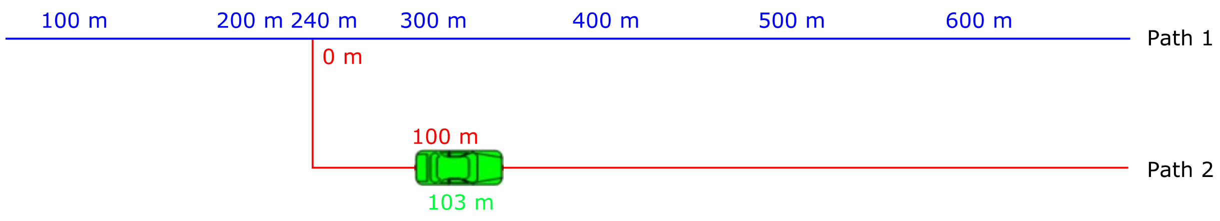


Figure 22 ADASIS v2 delete parent path (trailing length: 100m)

As it is shown in Figure 22, the parent path (path 1) will be deleted automatically when the vehicle is beyond the position path 2 offset 100m when the trailing length is 100m.

ADASIS v3 provides a more detailed mechanism that is fully under control of the provider. There is no extra logic needed on the receiver side.

Because paths themselves are managed through Path Control Messages and profile data on the paths through Profile Control Messages, separate life time definitions for the data on a path and the path object itself can be defined inside the provider. A simple example is given below with a road splitting in two branches at an intersection. Then the two branches are running in parallel for a long time.

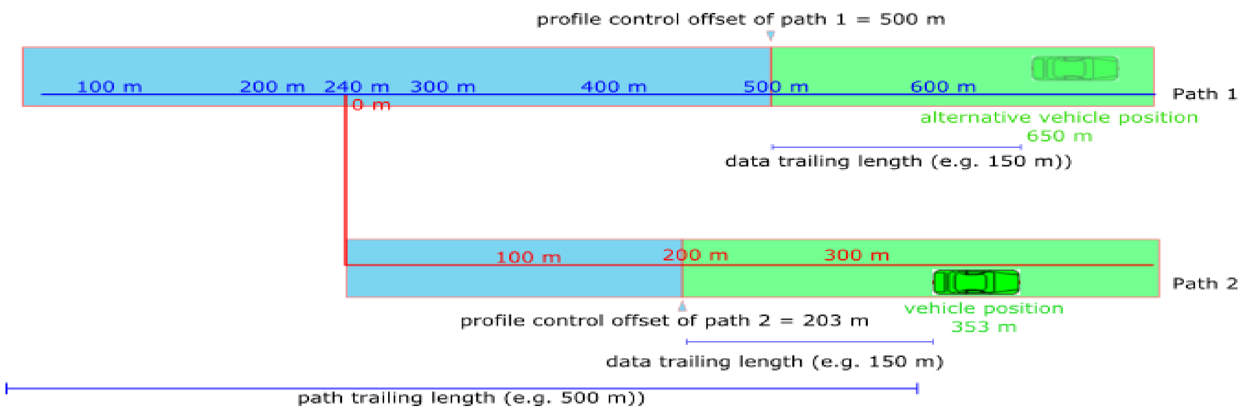

profile control offset of path 1 = 500 m
100 m
200 m
240 m
300 m
400 m
500 m
600 m
Path 1
0 m
alternative vehicle position
650 m
data trailing length (e.g. 150 m))
100 m
200 m
300 m
Path 2
profile control offset of path 2 = 203 m
vehicle position
353 m
data trailing length (e.g. 150 m)
path trailing length (e.g. 500 m))

Figure 23 ADASIS v3 delete parent path (path-data lifetime definition: 100m behind the vehicle, path lifetime definition: 500m behind the vehicle)

The provider can remove data on each path (parent and child) beyond the intersection point by sending the corresponding *Profile Control Offset* for each path and keep the parent path alive for a longer time by sending a path item within the *Path Control Message*. The green rectangles inside the picture illustrate the information of the *Profile Control Offset* for both paths. Inside those areas, data on the paths will exist. All data before the *Profile Control Offset* (=blue rectangles) can be deleted (path-data lifetime definition). The parent path (path 1) still exists even though the data at the beginning of the child path have been already deleted. New data inside the preview can be added on both paths (parent and child) till the provider decides something else.

One possibility could be to remove the parent path, if its map matching probability drops too low, for example. At this point, the provider only has to not reference the parent path ID when sending the next Path Control Message. This will trigger automatically the deletion of this path in all receivers (path lifetime definition).

In given example paths have a parent-child relationship. But this mechanism will work even for paths that are completely independent from each other (cf. parallel horizon trees).

Important:

- Sending a *Profile Control Offset* for a path means to inform the receivers up to which point data can be deleted on this path.
- Sending a Path Control item for a path in a *Path Control Message* means to keep this path alive at the receiver side.
  The structure of a Path Control Message is defined to transmit all currently managed paths and to transmit the parent-child relationship between the paths.

4.1.4 Dynamic lifetime calculation

As shown in the chapter “Separation between path life time and path-data life time” different lifetime definitions can be supported for the data and the logical object “path”. Because this logic has to be known only on the provider side, it is very easy to extend this mechanism. These definitions don’t have to be defined statically inside the provider. Finding and implement an algorithm, which allows the calculation of these lifetimes dynamically during a life cycle - maybe based on position probabilities -, is also possible.

4.2 Update and Erasure Mechanisms

In ADASIS v3, two update and erasure mechanisms exist for different purposes. It is possible to address and change a single profile value of any profile type. It is also possible to change an arbitrary number of profile values at once with a range update.

Both mechanisms use the *change* field of the Profile structure which selects one of the following operations:

- **Create**: New profile data is provided that should be added to the profile in the AHR.
- **Update**:
  - Previously transmitted profile data is updated. The AHR updates all changed fields to the new value.
  - Data can be removed by setting the availability flag of the profile struct to *NotAvailable*. This is used when profiles always require a value at each offset.
- **Delete**: Previously transmitted profile data is outdated. The AHR removes the data from the profile. The availability flag is ignored.

The following table summarizes all fields of a profile entry that can be updated depending on the profile interpolation type. All fields that are not mentioned in the table do not support updates.

	Spot		Step		Linear	
	Value	Range	Value	Range	Value	Range
confidence	●		●		●	
laneNumber	●		●			
offset	●		○**			
endOffset	●*		●*		●*	
endOffsetFinal***	●		●		●	
available	●	●	●	●	●	●
value	●	●	●	●	●	●

Legend

●	Update allowed
○	Update possible but not recommended
	Update forbidden

* Only recommended at the end of a profile to extend it
** too complex, use range update instead
*** endoffsetfinal only from false to true

All updates in a profile message should be applied in one step by the AHR. Otherwise applications may get inconsistent data when only parts of the updates in a profile message have been applied.

4.2.1 Value Update

A profile entry can be uniquely identified by a profile instance ID. This unique ID is provided by the AHP with the first transmission of the respective profile data to the AHR/application. This profile instance ID allows the AHR/application to easily identify the corresponding content to be adapted. The data change field indicates the action that is to be performed by the AHR/application on reception of existing content Traffic (traffic event profile values can be changed with a value update, for example.)

1. Example: Value update and removal

In the following example:

1. A ProfileMessage is transmitted to create a value on a profile.
2. Another ProfileMessage updates the value.
3. The last ProfileMessage removes the value from the profile. This creates a gap in the profile. Gaps prohibit error detection. If error detection is needed, the value should be updated to NotAvailable.

The example applies to profile values with step-, spot- or linear-interpolation.

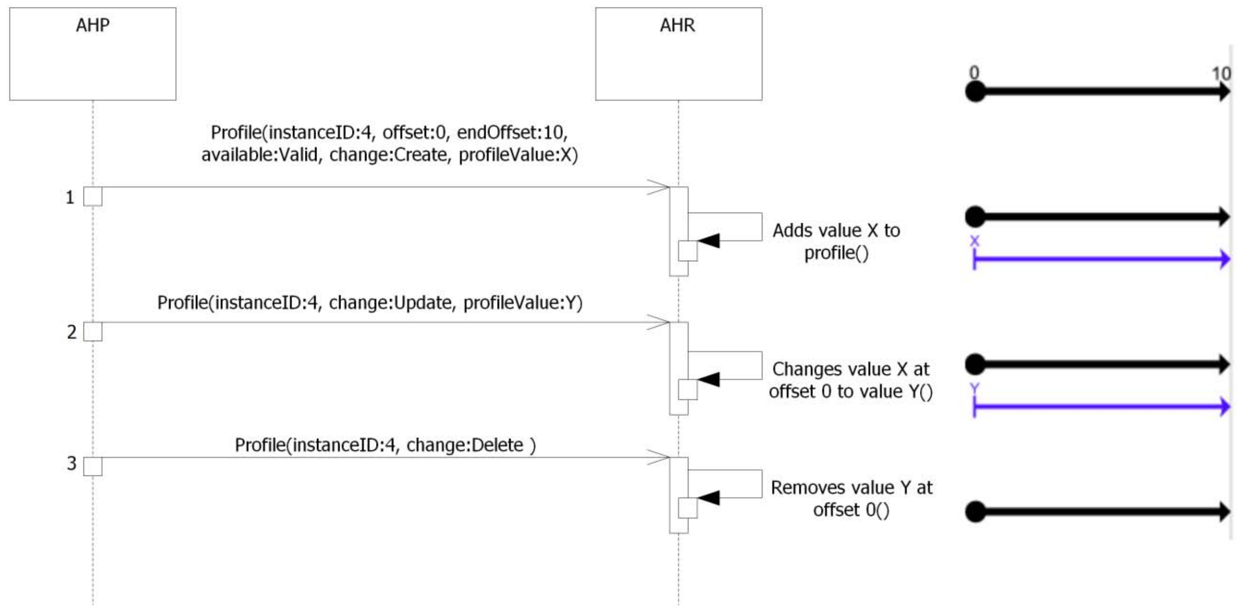

AHP
AHR
0
10
Profile(instanceID:4, offset:0, endOffset:10, available:Valid, change:Create, profileValue:X)
1
Adds value X to profile()
X
Profile(instanceID:4, change:Update, profileValue:Y)
2
Changes value X at offset 0 to value Y()
Y
Profile(instanceID:4, change:Delete)
3
Removes value Y at offset 0()

Figure 24 Value Update Overview

3. The profile value Y is replaced with NotAvailable. The profile is empty.

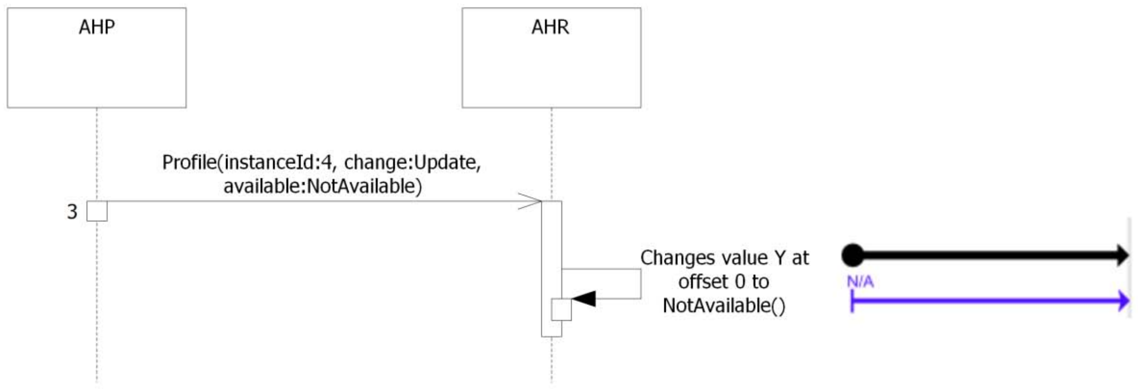

AHP
AHR
Profile(instanceId:4, change:Update, available:NotAvailable)
3
Changes value Y at offset 0 to NotAvailable()
N/A

Figure 25 Erasure by updating to NotAvailable

2. *Example: Updating the end offset at the end of a profile*

Updating the end offset is particularly useful when a path is extended by the provider but a profile value does not change. The end offset of the previous value can be updated to match the new length of the path.

The endOffsetFinal flag is updated to true to indicate that no further updates to endOffset will be received.

Alternatively, endOffset and endOffsetFinal can also both be updated in one message.

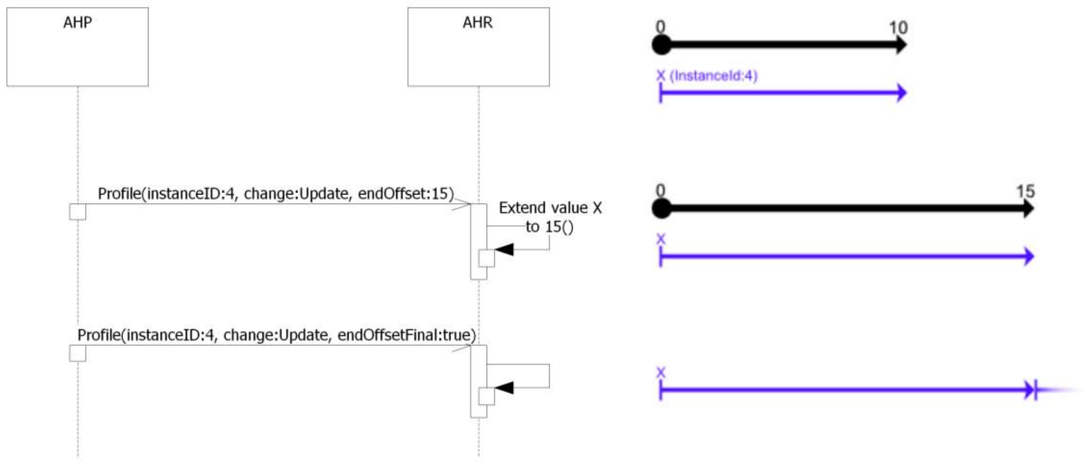

AHP
AHR
Profile(instanceID:4, change:Update, endOffset:15)
Extend value X to 15()
Profile(instanceID:4, change:Update, endOffsetFinal:true)
0
10
X (InstanceId:4)
0
15
X
X

Figure 26 Updating the endOffset

4.2.2 Range Update

The range update applies an operation to a whole offset range. Single values can't be updated with a range update. Therefore the unique instance ID 0 identifies an update operation as range update.

A range is addressed with the offset and endOffset fields of the profile entry structure. The update range is the interval [offset, endOffset), i.e., endOffset is exclusive. For the range update or range removal of profile data, the following rules apply:

1. The interval [offset, endOffset) must be aligned with existing data in the AHR. Otherwise it would be a partial delete or update which would result in inconsistent profiles. Therefore there must be profile value replacements for all values at offset and endOffset of an update or removal.
2. For range updates the order of operations in a ProfileMessage is important.
   a. Delete operations are always prepended in a ProfileMessage.
   b. Update operations for the deleted range follow.
3. To avoid gaps in profiles, delete and create operations of a range update shall be done in the same profile message

Traffic flow profile entries can be changed with a range update, for example.

1. Example: Full range update with one ProfileMessage

To update a range, all values are deleted. The new values are created with the same ProfileMessage and shall cover the same range.

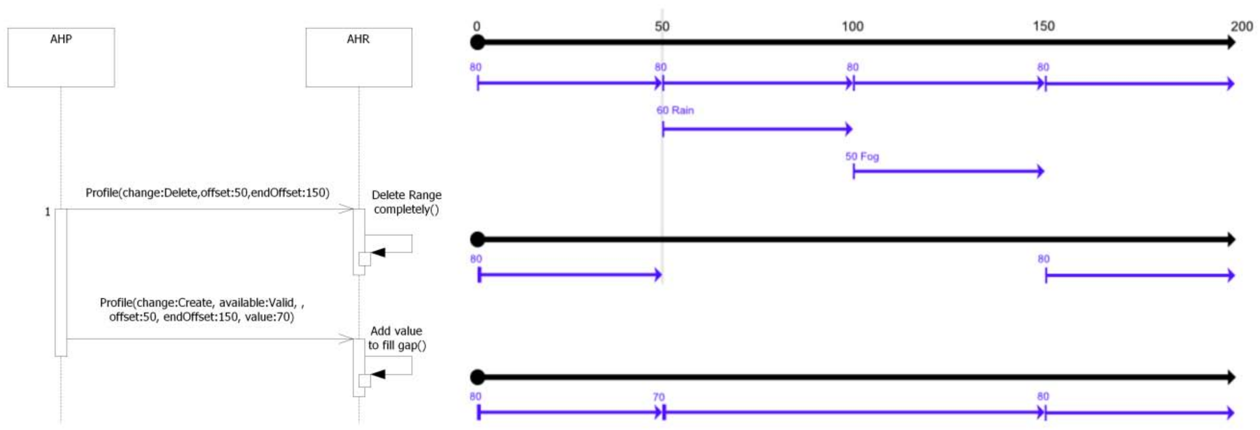

AHP
AHR
Profile(change:Delete,offset:50,endOffset:150)
Delete Range completely()
Profile(change:Create, available:Valid, , offset:50, endOffset:150, value:70)
Add value to fill gap()
0
50
100
150
200
80
60 Rain
50 Fog
70

Figure 27 Range update

2. *Example: Range invalidation*

Values in ranges can be removed at once with one of two methods. The example illustrates both, removing values in a range completely and updating a range to *NotAvailable*. The three operations are split among three ProfileMessages.

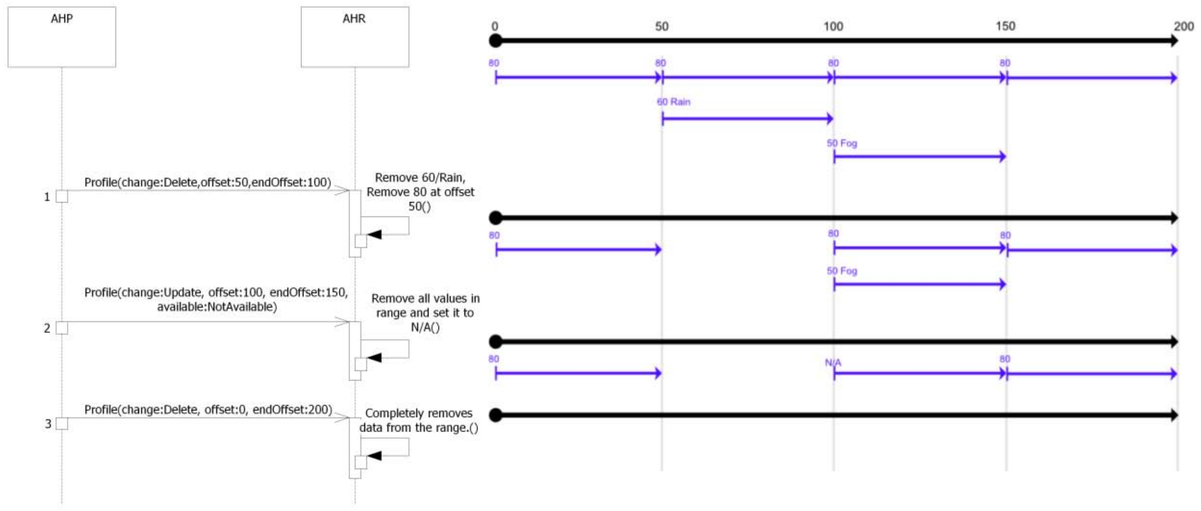

AHP
AHR
Profile(change:Delete,offset:50,endOffset:100)
Remove 60/Rain, Remove 80 at offset 50()
Profile(change:Update, offset:100, endOffset:150, available:NotAvailable)
Remove all values in range and set it to N/A()
Profile(change:Delete, offset:0, endOffset:200)
Completely removes data from the range.()
0
50
100
150
200
80
60 Rain
50 Fog
N/A

Figure 28 Range invalidation

1. The range delete results in two removed profile values and an empty range.
2. The range delete results in two removed profile values and a range with availability not available.
3. All values are removed from the profile at once, the profile is empty.

The example demonstrates the two possible options to remove ranges (1, 2). A provider can choose one of the options for a specific profile type.

4.3 [FORECAST] Memory Management

Size limits of each data type are predefined at the deployment and shared between provider and reconstructor. The reconstructor can create memory pools to save each data type. The provider will not send more data once the limit is reached for the specific data type. This means on the reconstructor side that the limits are the current stored elements plus the next message of the data type. As the provider shall not send more than these limits, from reconstructor perspective; these limits include the count of data stored and the next message from the provider.

In a standard ADASIS v3v3 environment, storage of data on the reconstructor side is controlled by the provider. The provider, of course, controls what data is collected by the reconstructor, just by sending the data. The provider also controls what data is deleted by the reconstructor through the Path Control and Profile Control messages. With this mechanism, the provider can know what data the reconstructor has in storage. However, this does not yet suffice for the provider to know whether the reconstructor has storage space left for another item of data - the provider also needs to know how much storage space the reconstructor has available in total.

With ADASISv3, the actual number of storage space sizes can be provided to both provider and reconstructor using Franca deployment definitions. But we also need a shared interpretation of these numbers.

In order to conform to requirements like those put forward by MISRA (prohibiting type casting), we use a type-safe allocation model. For each data type that might be allocated dynamically, we have a storage pool which has a fixed number of items available. Data items are allocated from the pool corresponding to their data type, and returned to this pool. The number of items in the pool can be given in the Franca deployment definition per data type.

A crucial term, though, in this description is "dynamically allocated". Not every data item in the ADASISv3 horizon needs to be dynamically allocated by the reconstructor. To be specific:

1. A data item does not need to be allocated dynamically if it is a direct member of a structure.
2. For an array (except for inline arrays, see below), each array member needs to be allocated dynamically.
3. For any polymorphic data item, an instance (having a specific data type) needs to be allocated dynamically from the pool for the specific data type.

Please note that to avoid storage pool fragmentation issues, Franca arrays might be stored internally as linked lists.

With this approach, every dynamically allocated item (in particular instances of a polymorphic type) could uniquely be identified by naming a data type and providing an index into the storage pool for this data type.

As a special case, small arrays might be inlined. This means allocating memory for the maximum possible number of array entries inside the enclosing data structure instead of dynamically allocating the individual array entries from a pool. This, of course, requires a maximum size to be defined for the array in question which can be done via the MaxSize deployment property defined in the standard ADASIS v3 Franca deployment specification. On top of that, the array should be marked for inlining, this is yet TBD.

Now with this way of counting, we can give for each structure data type a limit to the number of instances existing at the same time (that is, a storage pool size) in the "MaxOccurences" deployment property of that structure data type. This deployment property is specified in the standard ADASIS v3 Franca deployment specification; the actual values can be specified by implementers in their deployment definitions.

Up to now, there is no equivalent facility for non-structure data types (scalar types, enumerations) that might need to be allocated dynamically. The expectation is that arrays of these data types generally will be inlined, due to the large management overhead when allocating them dynamically. Otherwise, it would be easy to specify a corresponding deployment property for enumerations while for built-in data types, global deployment properties of the Franca type collection would be needed.

Having both the way of counting and the actual limits, we finally can define what to count: We take a count over all profile data that should currently be in the ADASIS v3 horizon, i.e. assuming that all Path Control messages have been processed to delete outdated paths, and that all Profile Control messages have been processed to delete outdated profile entries on existing paths. In addition, for global data we include the latest value for each profile type, and we include the position data from the latest position message. All this data, and on top of that the content of a new message created by the ADASIS v3 Horizon Provider, has to stay within the storage pool size limits. In other words, both the existing horizon content and the next message to be processed have to jointly fit into the storage available to the reconstructor - this rule enables (but does not require) allocating storage for a message being decoded from the same storage pools as the horizon content.

Please note that this description does not require an implementation to literally follow the approach described here. Implementations may manage their data in whatever way, just the capabilities of a reconstructor to store and process data have to conform at least to what is described here; and a provider must a most send as much data as described here.

4.4 [FORECAST] AHR Feedback

The AHR/application has the possibility to provide informative feedback on delivered profile data toward the AHP using the same profile instance ID. Similar to the communication from AHP to AHR, the AHR/application can indicate in its feedback message which adaptation the AHP is supposed to perform on the provided data. For this purpose, the AHR/application shall use the data change field in the profile data to provide the AHP with relevant feedback on the delivered profile data. For instance, the AHR/application can approve certain profile data, e.g. existence or validity of speed limits. Further, it may request to correct certain profile data and provide its own observed value toward the AHP. Furthermore, the AHR/application may request to add newly detected profile data to the AHP database which can approve or disprove data.

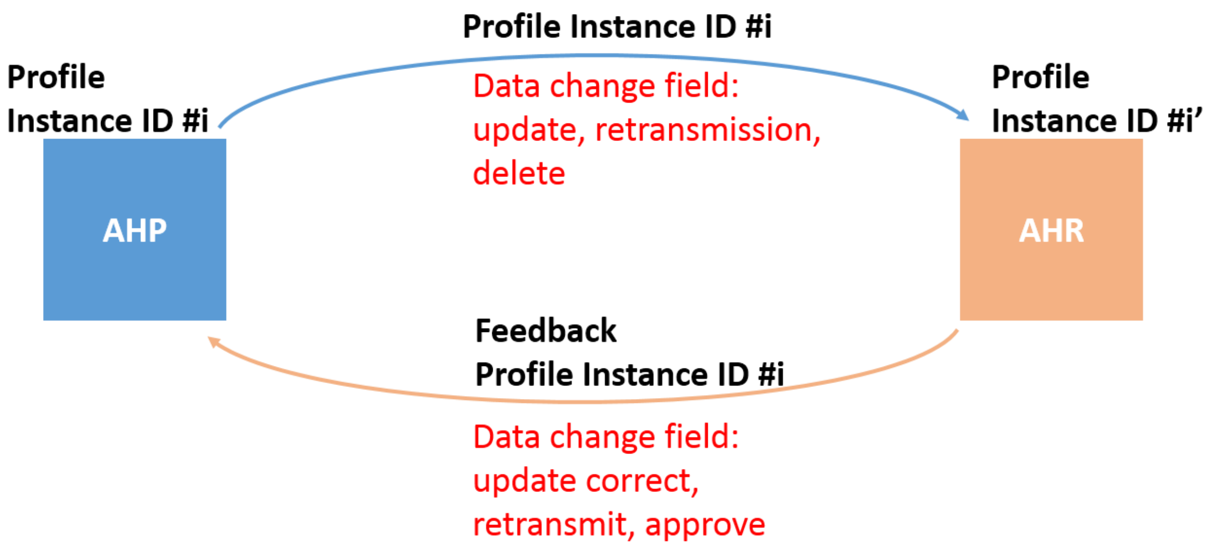

Profile Instance ID #i
Profile Instance ID #i
Data change field: update, retransmission, delete
Profile Instance ID #i'
AHP
AHR
Feedback Profile Instance ID #i
Data change field: update correct, retransmit, approve

Figure 29 EHR Feedback

4.5 Coexistence with ADASIS v2

Due to evolutionary changes in vehicle architectures, often both ADASIS v2 and ADASIS v3 based application will exist inside a single vehicle, requiring both an ADASIS v2 and an ADASIS v3 Horizon provider.

These providers can coexist independently of each other. By default ADASIS v2 is transmitted via a CAN bus, while ADASIS v3 does not support CAN and needs another transport medium. But even if ADASIS v2 is also sent via a non-CAN medium, it can be encapsulated in a way that distinguishes ADASIS v2 from ADASIS v3 and thus allows fully independent transmission of both protocols.

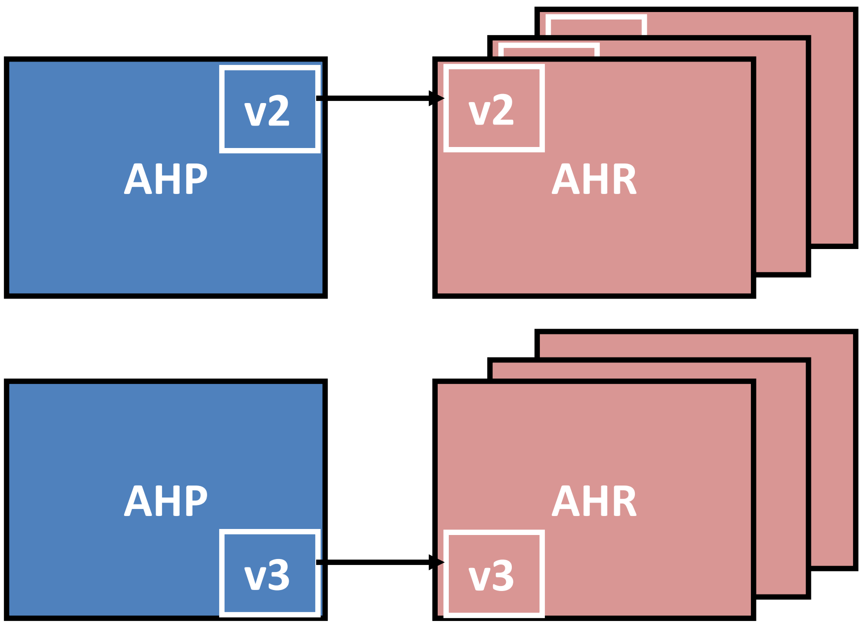


Figure 30 Coexistence Scenario of ADASIS v2 and v3 Horizon Providers

It might be advantageous, though, not to do two fully separate horizon provider implementations. Rather, one could save some overhead of horizon building and construct an ADASIS Horizon only once inside a single provider. This provider would generate both ADASIS v2 and ADASIS v3 messages from the same internal ADASIS Horizon data structure and would implement two different interfaces for providing ADASIS v2 and v3 messages. Besides saving computational efforts for constructing the ADASIS Horizon, this has the additional advantage that the ADASIS v2 and ADASIS v3 horizons will be synchronized, i.e., the MPP will be the same for both.

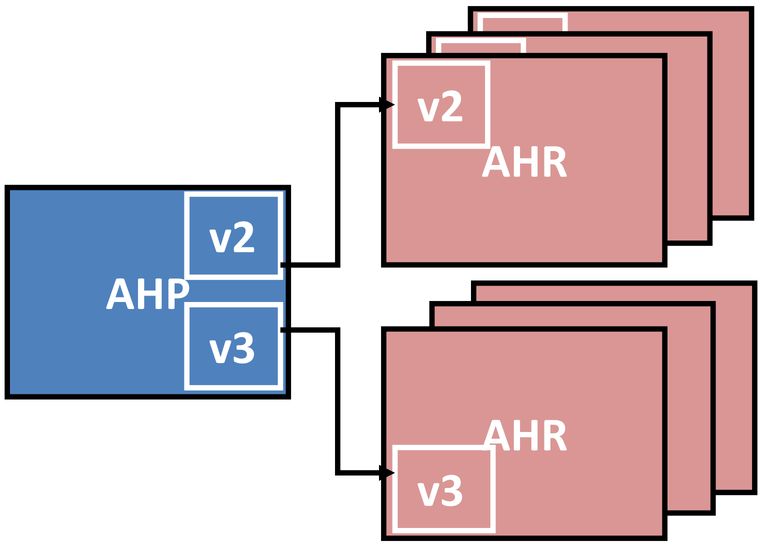


Figure 31 Single AHP Providing ADASIS v2 and v3 Messages

On the other hand, it is hardly feasible to have only an ADASIS v3 Horizon provider and then a converter translating ADASIS v3 to ADASIS v2. Such a conversion usually is not possible at the message level; instead a converter would need to include an ADASIS v3 Horizon reconstructor and a full ADASIS v2 Horizon provider. In principle this would be possible, but the more appropriate solution might be a combined ADASIS v2 and ADASIS v3 provider as described above, if necessary transmitting ADASIS v2 encapsulated over some non-CAN network where a converter can unpack it and forward it via CAN.

4.6 [FORECAST] Auxiliary Data Providers and Sensor Fusion

In ADASIS v3, it is not required that all ADASIS v3 Horizon data is supplied by one single horizon provider. Rather, as depicted in Figure 32, it is also possible that a core ADASIS v3 Horizon provider supplies the topology of the road network (using path control messages) and some of the attributes (i.e., profile data). In addition, an auxiliary horizon data provider could supply additional profile data, creating profile messages that use the paths defined by the core horizon provider as a system of reference.

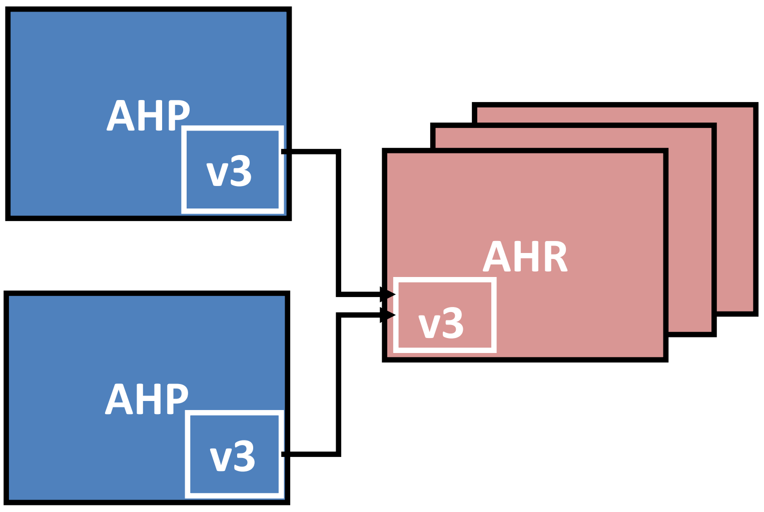


Figure 32 System Diagram of Multiple (Core and Auxiliary) ADASIS v3 Horizon Providers

Having different providers populating the ADAS Horizon is thus the key concept enabling to convey a wide range of information. Similar to ADASIS v2, the core provider sends the road topology as described in a DB. But in ADASIS v3, this information is only the skeleton that can be populated by other providers. Each of them can specifically provide additional information coming from sensors or sensor fusion units.

As we see today, vehicles are becoming more and more capable of performing highly automated driving and will soon reach the ultimate stage of the autonomous car. In that context, it is clear that vehicle controllers require accurate and reliable information about their immediate environment, along with the short and long-range predictions of the road ahead. Provision of such reliable and accurate information is the aim of sensor fusion. Sensor fusion is all about aggregating complementary or redundant information coming from a network of sensors in order to enhance system capability and reliability. Quality and covering of sensor fusion output is thus a strong enabler for the provision of algorithms that are able to cope with the wide range of situations that arise in autonomous driving. Sensor fusion is a hot topic that is addressed by a large research community and many publications, both from academia and the industrial world. The sensor fusion architecture ultimately depends on OEM needs, on the level of autonomy and on the type of involved sensors.

Sensor fusion is a subject well beyond the scope of ADASIS v3. However, ADASIS v3 must support the sensor fusion process by being flexible enough to cope with different kinds of fusion architectures. For that reason, ADASIS v3 shall not be only able to provide “static” information about the road ahead, such as the information coming from a database. It must also be able to convey “dynamic” information coming from other sources, e.g., vehicle sensors.

ADASIS v3 doesn’t deal with sensor fusion directly, but support many sensor fusion architectures through its flexibility.

The concept of multiple ADASIS v3 horizon providers is illustrated in Figure 33. This concept supports fusion of sensor data from various sources inside the system (for example, speed limit information both from a map and from camera based detection). Each data source can use the ADASIS v3 protocol to send its own data. Therefore, each data source listens to the ADAS horizon sent by the core provider and then enriches it with any additional information. Each provider can be arbitrarily defined and has a specific responsibility in terms of provision and update of specific profile data. The only caveat is that only a single data provider may send a certain profile type inside an ADASIS v3 system, so providers of unfused sensor data will need to use custom profile types of data corresponding to a standard profile type that would be used for the fused data.

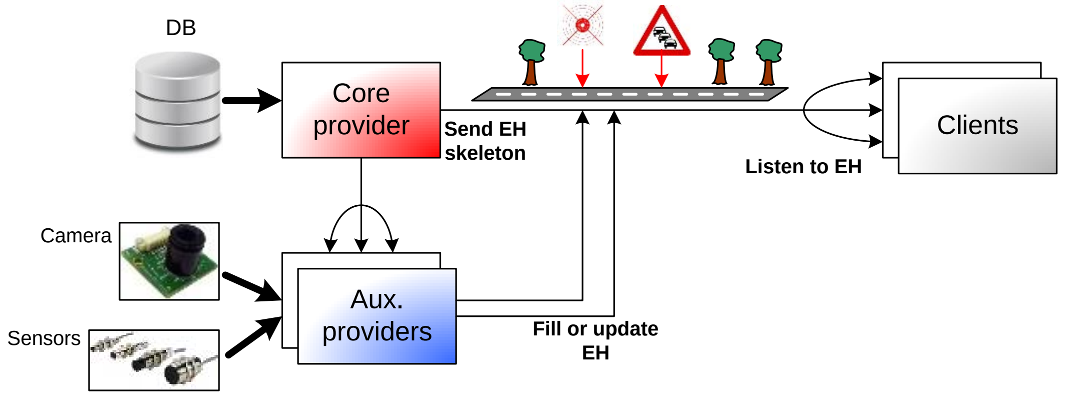


Figure 33 Concept of Multiple (Core and Auxiliary) ADASIS v3 Horizon Providers

We can imagine several different architectures thanks to this concept. For example, the applications listening to the ADAS horizon could gather multiple data sources and perform their own data fusion, or a central instance can handle data fusion and supply the fused data – again via ADASIS v3 – to the applications

As stated above, algorithms controlling the vehicle need reliable sensor fusion output. However there are two different extreme ways of using ADASIS for supporting the sensor fusion process: downstream and upstream fusion. The two examples described below illustrate these two antipodal concepts. Of course, a real architecture using ADASIS v3 is likely to position itself somewhere in between.

Downstream fusion

The first way to perform sensor fusion is conservative and uses ADASIS v3 in a way similar to ADASIS v2. The map is a specific kind of sensor and the ADAS Horizon is seen as a means to broadcast map information across the vehicle. In addition to map information, ADASIS v3 can here be used to broadcast information coming from other sensors, thanks to additional "providers". As a result, sensor fusion is a process that is done at the client side, where the pre-processed sensor data is perceived, hence the name downstream fusion.

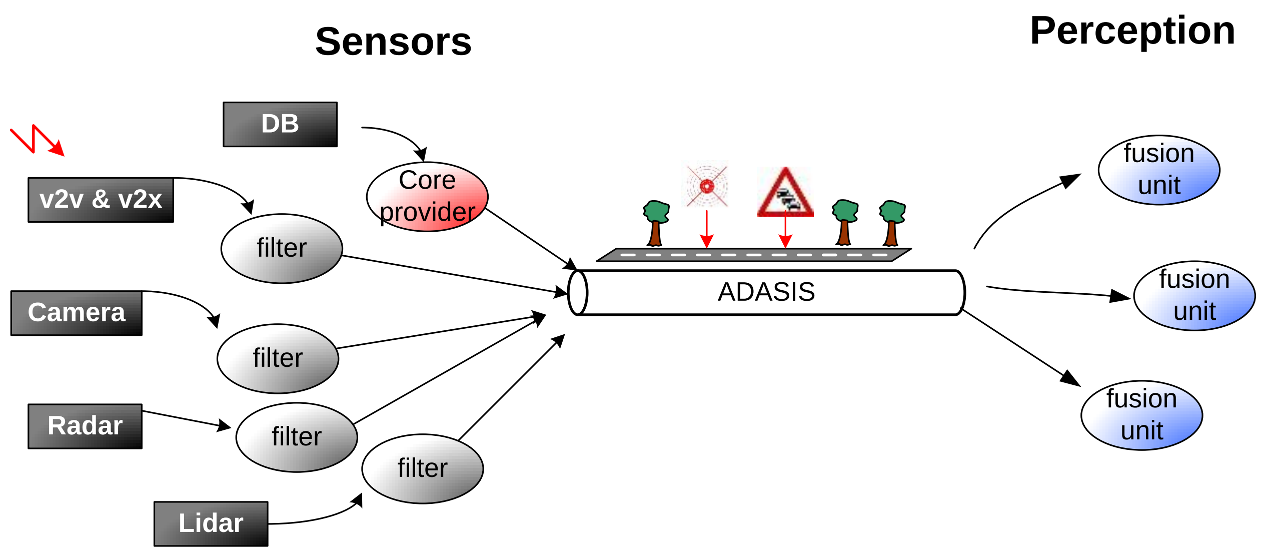


Figure 34 Downstream Sensor Fusion Concept

Upstream fusion

The second way to perform sensor fusion considers that ADASIS v3 is a means to spread consistent and accurate knowledge of the road ahead and the surrounding world, directly suitable for autonomous vehicle control algorithms. Thus, the sensor fusion process is handled as closely as possible to the raw sensor data at the provider side, hence the name upstream fusion, and the outputs of different sensor fusion units are used to feed and enrich the AH.

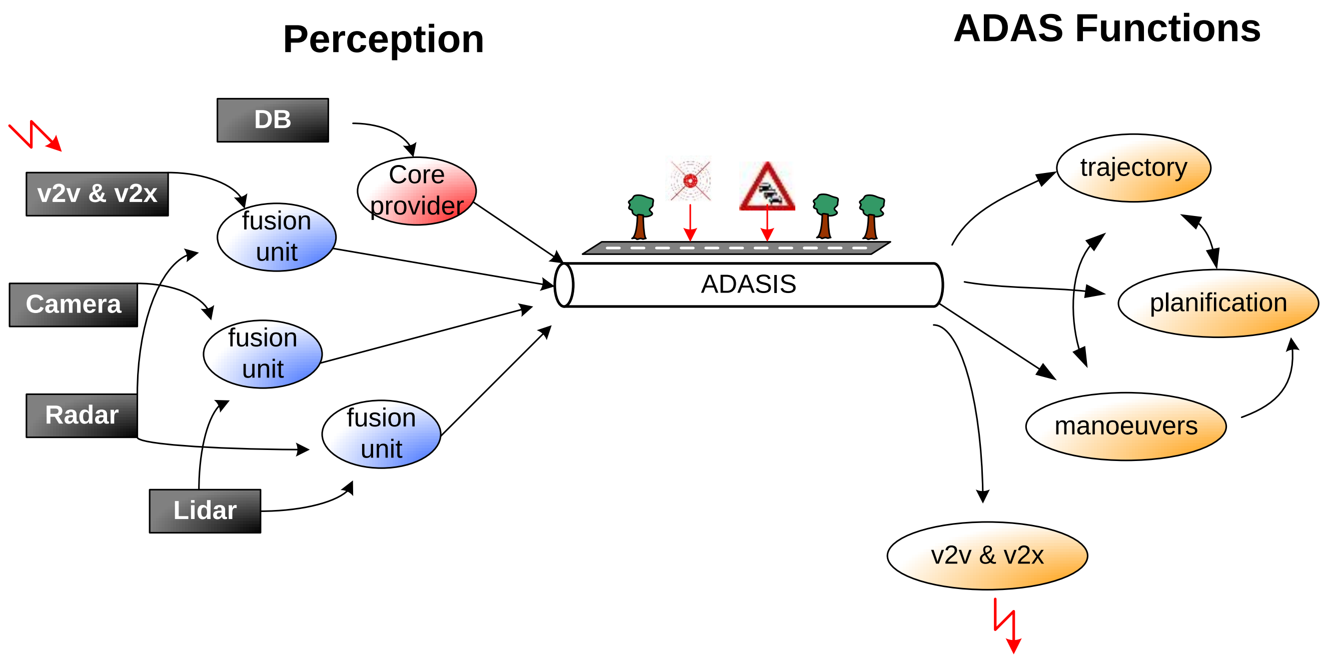


Figure 35 Upstream Sensor Fusion Concept

4.7 Communication Schemes

For the communication between AHP and AHR, the following schemes are supported:

- Broadcast
- Request/provide (on demand query)
- Subscribe/publish (e.g., time or event based)

The current version of ADASIS v3 deals with Broadcast mechanism **only**. Request/provide and Subscribe/publish are the focus of future versions.

The general information flow between ADASIS v3 entities is depicted in Figure 36.

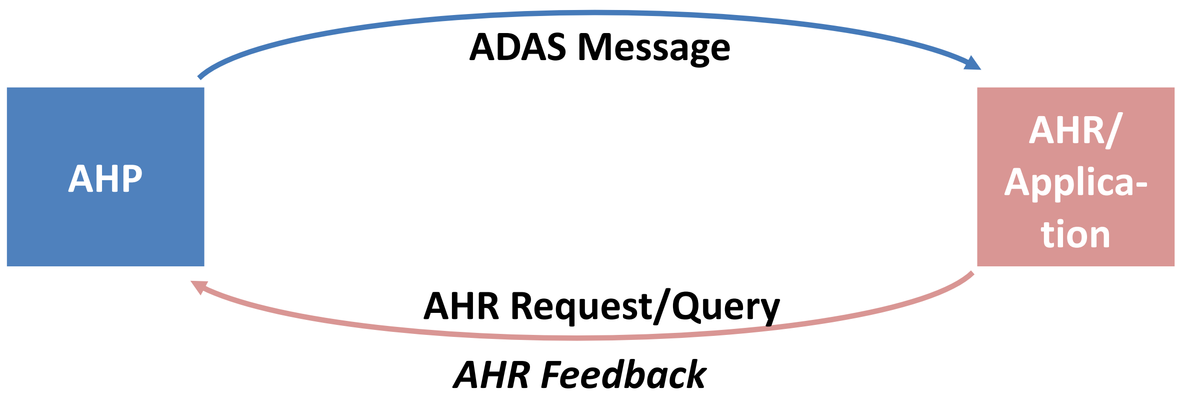


Figure 36 General ADASIS v3 Information Flow

In addition to information provisioning from AHP to the AHR/application, the application can explicitly request or subscribe for specific profile data. Further, the AHR/application shall be able to provide the AHP with informative feedback on delivered data. This allows for approving or correcting received profile data, notifying the AHP of data changes or newly detected information and enables the AHP to improve its database and performance.

4.7.1 Broadcast

In the simplest case, an ADASIS v3 Horizon provider creates a single message stream and distributes these messages to all its client applications. This can be achieved by using a pure broadcast medium (as is the CAN bus with ADASIS v2), by using a broadcast or multicast feature of the transport protocol, or by maintaining point-to-point connections to the clients and sending identical data via all connections. From the ADASIS v3 perspective, all these options constitute broadcast network topologies. Only the descriptions of how to implement ADASIS v3 on top of some given transport protocol will specify details of the physical topology.

Further, it is possible for a client to request data from an ADASIS v3 Horizon provider even with an ADASIS v3 broadcast topology, removing the need for manual configuration. In this case, the provider needs to send the union of the data requested by all clients. Clients might have to filter out data that they do not require. While filtering is easily possible on the level of the profile, care must be taken when filtering by distance from the vehicle or sub-path level. In broadcast mode, a provider will usually send data only once, and a client must store it until it is needed.

4.7.2 [FORECAST] Multicast

[ToDO]

4.7.3 [FORECAST] Unicast

[ToDO]

4.7.4 [FORECAST] Request/Response

In the case of ADASIS v3 broadcast, it is feasible to manually configure the details of what an ADASIS v3 Horizon provider should send (this will usually be a superset of all the data required by the individual clients). If the ADASIS v3 Horizon provider maintains separate point-to-point connections to the clients, it is also possible to generate a tailored data stream for each individual client. In case of tailored data streams, manual configuration is theoretically still possible, but it will be cumbersome and error-prone. In general, the preferred solution is that each client informs the provider of its data needs. ADASIS v3 defines messages (see below) for a client to send this information and request data from the provider.

The actual RequestMessage is a message from a client to the provider. It does intentionally not contain an identity of the sending client – this cannot be specified in a transport protocol independent way. It is expected that a transport protocol can convey a sender identity or address of the client that later can be used by the provider to answer to the client. If a transport protocol should fail this expectation, then the specification of the encapsulation of ADASIS v3 over this transport needs to take care of handling client identities.

The RequestMessage contains a list of RequestItem entries. Each RequestItem can be a Subscribe, Unsubscribe, or SendOnce request. The Subscribe and Unsubscribe requests are typically used in a subscribe/publish scheme. In a simple request/response scheme, a SendOnce request causes a one-time sending of data corresponding to the request. It may also be used to request sending of "old" data not (re-)sent after a Subscribe request.

A RequestItem specifies a MessageType the client is interested in. For Profile and GlobalData messages, one ProfileType needs to be specified. For Profile messages only, the client specifies in which part of the ADASIS Horizon it is interested in using the sidePathLevel, the notDrivablePaths flag, and the distance from the vehicle position in centimeters. For the sidePathLevel, a value of 0 asks for the vehicle path only, 1 asks for first level sub-paths of the vehicle path as well, etc. As a special case, a value of 255 asks for all sub-paths. The notDrivablePaths flag decides whether paths are needed as well as that the ego vehicle cannot drive to, e.g., one-way roads in the wrong direction. (Knowledge of these paths might be useful to understand the movements of other vehicles.) Please note that the distance is measured from the current vehicle position even for sub-paths (not from the start of the path).

4.7.5 [FORECAST] Publish/ Subscribe

An AHR/application can subscribe at the AHP for specific ADAS profile data, where conditions for data delivery are to be included in a RequestMessage issued by the AHR toward the AHP. This communication scheme enables applications to receive updated profile data in case pre-defined conditions are satisfied, e.g., time or event based profile updates. For example, a Subscribe request asks the provider to send some item whenever it is added to the provider internal ADASIS Horizon in the future. It does not imply to send again all the corresponding data that have been sent before the subscription. An Unsubscribe request cancels a previous subscription.

The subscribe/publish scheme requires the AHP to maintain a registry of applications and active subscriptions. Further, multiple subscriptions of a single client will frequently overlap. In this case, the provider sends the union of all data that corresponds to the current subscriptions of a client.

5 Use of ADASIS v3 Protocol

5.1 ADASIS v3 Messages

Data transmission in ADASIS v3 is based on a message concept. These messages are a kind of a container which packs data into structured format. Different types of message could be used to add different transmission relevant information to each contained data and to transmit them in several ways, for example on different channels with different timings.

As a result the type of its content is known before unpacking and can also be handled in different ways if needed.

ADASIS v3 has four different types of messages:

1. **Synchronization Messages**, which give *instructions* which paths and profiles are created, changed or deleted.
2. **Position Message**, which transmits *positioning* data of the vehicle.
3. **Global Data Message**, which provides global information which is *not linked to any path*.
4. **Profile Message** contains all possible *profiles linked to a path* and describing the horizon.

Each message and its members are described in the reference document ([2]).

5.2 Profiles

ADASIS v3 offers a rich set of standardized profiles which are summarily listed below. Furthermore, custom profiles can be added as needed for a specific implementation.

Each profile type is described in the reference document ([2]).

5.2.1 Custom Profiles

It is easily possible to extend ADASIS v3 for transmitting custom profiles. Basically anything that can be attached to a location on the road network can be sent in a custom profile.

The process is rather simple, and an example is provided together with the ADASIS v3 Franca IDL files and code generator sources (files ADASISv3MyCustomized.*):

- Create your own .fidl and .fdepl files from these samples. Please change the file names, and most importantly change the namespace, so you do not in the future get conflicts with other custom extensions.
- Define an enumeration assigning an ID to your custom profile. Logically this is an extension of the standard ProfileType enumeration, so IDs should be manually chosen not to conflict with that.
- If your custom profile does not use one of the existing data types, then define a structure containing the data you need.
- In your custom .fdepl file, take over the data from ADASISv3.fdepl, as shown in the sample, then define the interpolation type for your custom profile, and assign a tag for your custom data type (if you use a custom data type).
- Implement your provider to send your custom profile, using the ID and the data type you defined.
- A reconstructor should just need recompilation using your extension .fidl and .fdepl files.
- Applications now should be able to query the reconstructor for your custom profile and to get back data with the data type you have defined.

6 Guidelines and Recommendations

6.1 Bandwidth Saving

ADASIS v3 can provide a lot of information and logic, which can be used inside a vehicle. A problem can be that many of the current ECUs can't provide all features at the moment because of the limitations of their hardware (CPU, memory). The amount of data is always increasing but the current hardware can't be changed. Therefore a few mechanisms to decrease the amount data should be possible.

In this part of the synchronization of the data between provider and receiver this possibility is given by defining a maximum number of paths, which are allowed to exist at the same time. With this maximum amount of paths the Profile Control Message structure will be filled maximum with this number of items.

6.2 Implementation recommendations

6.2.1 Number of paths

ADASIS v3 doesn't define a maximum number of paths which build up a Horizon tree, but recommends allocating up to 56 paths simultaneously, including the current segment on which the vehicle is positioned.

6.2.2 Message with identical profile type at same offset

If you need to send

- several profiles
- of same type
- at same offset

you have to send them in <u>same profile message</u>.

In that way you guarantee consistency check. An example for such use case could be several traffic signs at same offset. Another example could be several extended speed limits. But please keep in mind: Overlapping of profiles is not allowed! In such case you must split a profile to ensure common offset.

7 Error Detection and Recovery

ADASIS v3 aims at providing a consistent data stream for ADAS applications, no matter whether only one or multiple applications are receiving the same data stream.

In order to enable proper and stable overall system operations, the ADASIS v3 protocol exhibits several means for error detection and recovery, which are described in the following sections.

7.1 Error Detection

ADASIS v3 allows for detecting missing, lost, or reordered messages as well as identifying gaps in transmitted horizon data.

7.1.1 Missing or Reordered Messages

Each ADASIS v3 message includes a cyclic counter that increases with a new message generation. Thus, a receiver can check whether ADASIS v3 messages are received in order, out of order, or are not received at all (e.g., messages are lost due to issues on transmission medium).

An ADASIS v3 receiver can check whether subsequently received messages exhibit incrementally increasing cyclic counters as illustrated below:

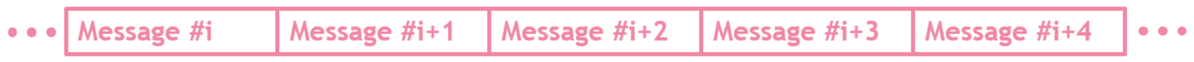


Figure 37 Message specific cyclic counter increases with every new message

If single or multiple messages are not received (or lost) the sequence of incrementally increasing cyclic counters will exhibit a “hole”, i.e., the sequence of cyclic counter values is not continuously increasing but interrupted due to missing counter values. This is exemplarily depicted below:

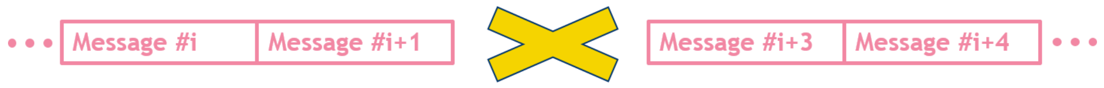


Figure 38 Missing or lost message identified due to interrupted sequence of cyclic counter values

Further, situations where the horizon provider decides to delay or reschedule the transmission of a particular message to a later point in time may also occur. This could, for example, happen if specific horizon data is not available at foreseen transmission time or data delivery to the horizon provider (e.g., from external services) is delayed.

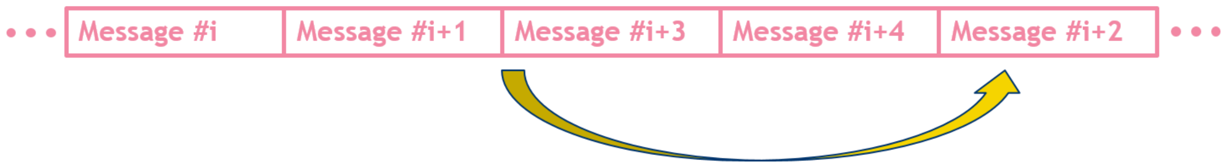


Figure 39 Reordered message sequence

7.1.2 Gap Detection

For detecting gaps in horizon data (e.g., resulting from map failure or message transmission issues) the concept of validity length can be used. The validity length information is part of each profile entry and consists of the following tuple (offset, endOffset, endOffsetFinal).

In case of continuous profile data the validity length interval of one entry ends with an endOffset that matches the profile range offset, i.e., start of validity range interval, of the following profile entry of the same type as illustrated below:

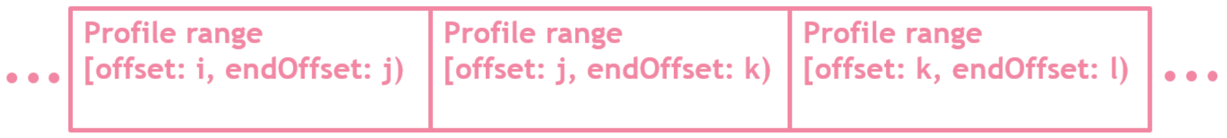

• • •
Profile range
[offset: i, endOffset: j)
Profile range
[offset: j, endOffset: k)
Profile range
[offset: k, endOffset: l)
• • •

Figure 40 Profile data include validity length information based on offset and endOffset

In case of profile data gaps, the endOffset of one profile entry does not coincide with the profile range offset of the following profile entry of that type. Thus, checking the profile ranges or validity length interval of subsequent profile entries of the same type enables gap detection and completeness check.

The situation of mismatching validity length intervals between subsequent profile entries of the same type is exemplarily depicted below:

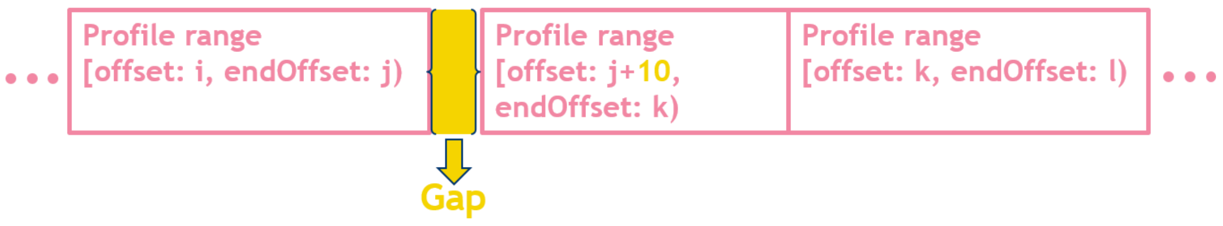


Figure 41 Profile data range check allows for gap detection

7.2 Error Recovery

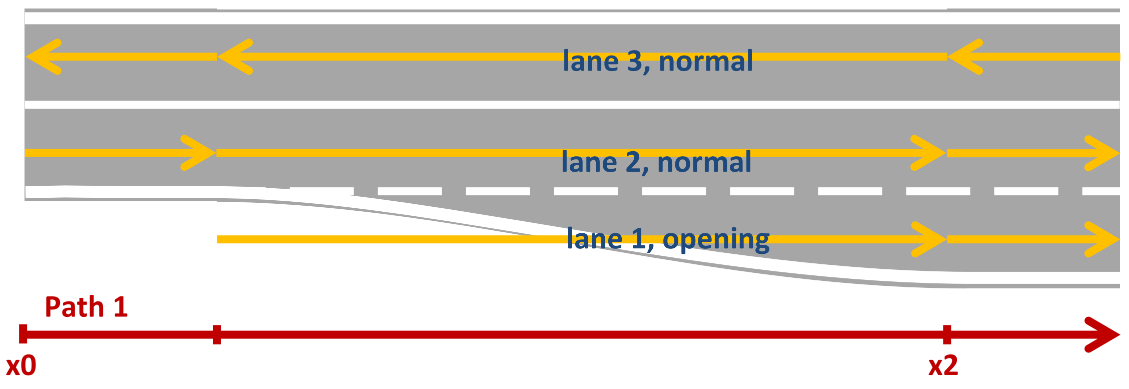
As there is no reverse channel (from receivers to providers) in the first ADASIS v3 release, receivers do not have means for providing feedback on message transmission related issues to the horizon provider.

Thus, receivers have to wait for message retransmissions the provider may send depending on its configuration.

8 Examples of Use

8.1 Lane Connectivity

8.1.1 Standard case of a lane forming over a significant distance

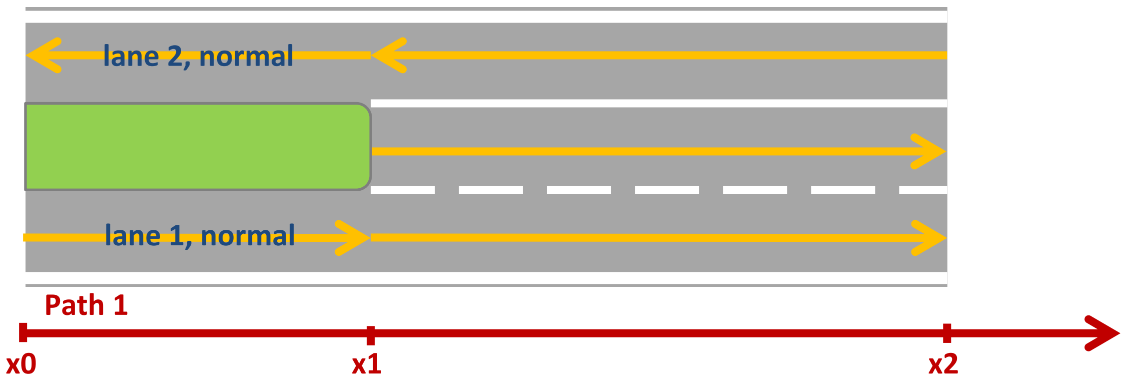


- **Path 1, offset x0, segment of length (x1-x0):** 2 lanes
  - lane 1: in driving direction | Transition: none
  - lane 2: against driving direction | Transition: none

- **Path 1, offset x1, connection node:**
  lane connections
    - lane 1 -> lane 2
      lane 2 -> lane 3
- **Path 1, offset x1, segment of length (x2 – x1):** 3 lanes
    - lane 1: in driving direction | Transition: opening
    - lane 2: in driving direction | Transition: none
    - lane 3: against driving direction | Transition: none
- **Path1, offset x2, connection node:**
  lane connections
    - lane 1 -> lane 1
    - lane 2 -> lane 2
    - lane 3 -> lane 3
- **Path 1, offset x2, segment …:** 3 lanes
    - lane 1: in driving direction | Transition: none
    - lane 2: in driving direction | Transition: none
    - lane 3: against driving direction | Transition: none

Hint: Please note that since the new lane forms only after offset x1, it's not possible to change directly to it at x1, rather this requires a lane change maneuver between x1 and x2. There is no connectivity at x1 between lane 1(x0) to the opening lane (lane 1(x1)).

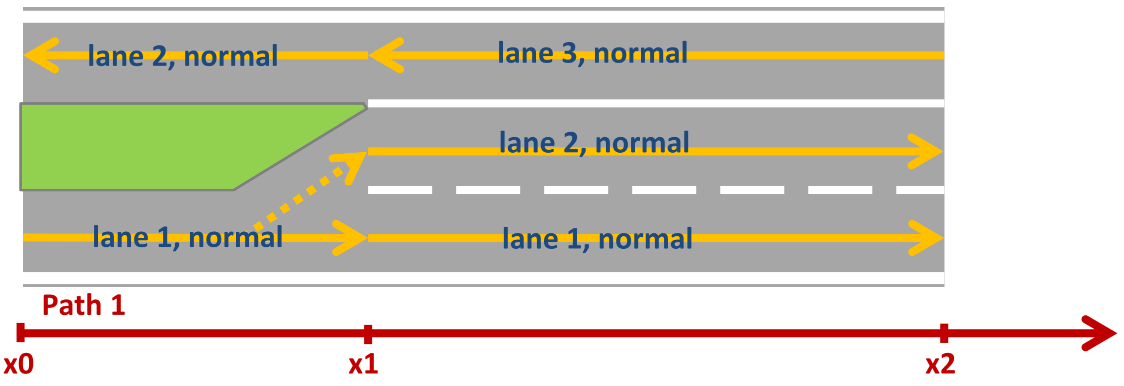
8.1.2 Special case of a lane forming abruptly



- **Path 1, offset x0, segment of length (x1-x0):** 2 lanes
    - lane 1: in driving direction | Transition: none
    - lane 2: against driving direction | Transition: none
- **Path 1, offset x1, connection node:**
  lane connections
    - lane 1 -> lane 1
    - lane 2 -> lane 3
- **Path 1, offset x1, segment of length (x2 – x1)**: 3 lanes
  lane 1: in driving direction | Transition: none
  lane 2: in driving direction | Transition: none
  lane 3: against driving direction | Transition: none

Hint: There is no connectivity at x1 to lane 2 (x1). A lane change maneuver is required to get into the new lane.

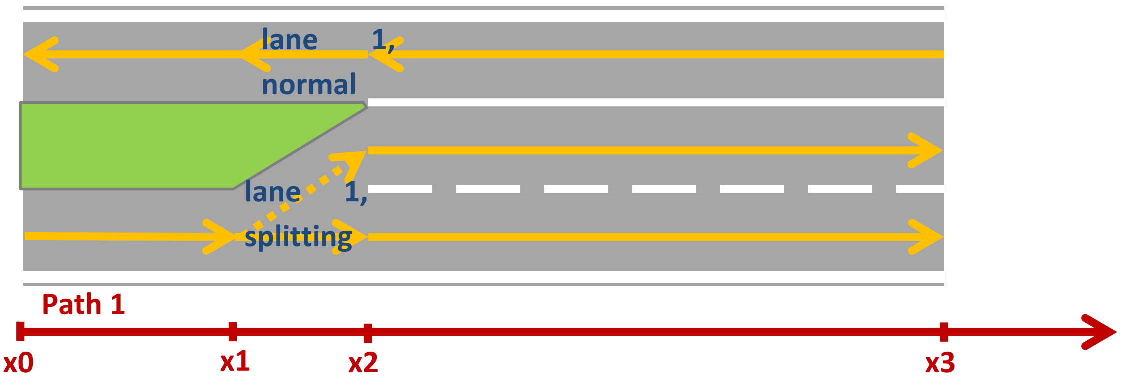
8.1.3 Special case of a lane splitting



- **Path 1, offset x0, segment of length (x1-x0):** 2 lanes
  - lane 1: in driving direction | Transition: none
  - lane 2: against driving direction | Transition: none
- **Path 1, offset x1, connection node:**
  lane connections
  - lane 1 -> lane 1
  - lane 1 -> lane 2
  - lane 2 -> lane 3
- **Path 1, offset x1, segment of length (x2 – x1)**: 3 lanes
  - lane 1: in driving direction | Transition: none
  - lane 2: in driving direction | Transition: none
  - lane 3: against driving direction | Transition: none

Hint: In this case the road markings suggest that the driver can choose to use the new lane from its start point on. In this example the few meters of *widening* of the lane is not modeled as an *opening or splitting transition*.
Difference to former example: lane 1 (x0) has two successors.

Alternatively, if the widening should be modeled as a transition then a further road segment has to be introduced between x0 and x1.



- **Path 1, offset x1, segment of length (x2-x1):** 2 lanes
  - lane 1: in driving direction | Transition: splitting
  - lane 2: against driving direction | Transition: none

It is expected that such different view also depends on digitization of data sources (map).

8.2 Different levels of details in a T intersection

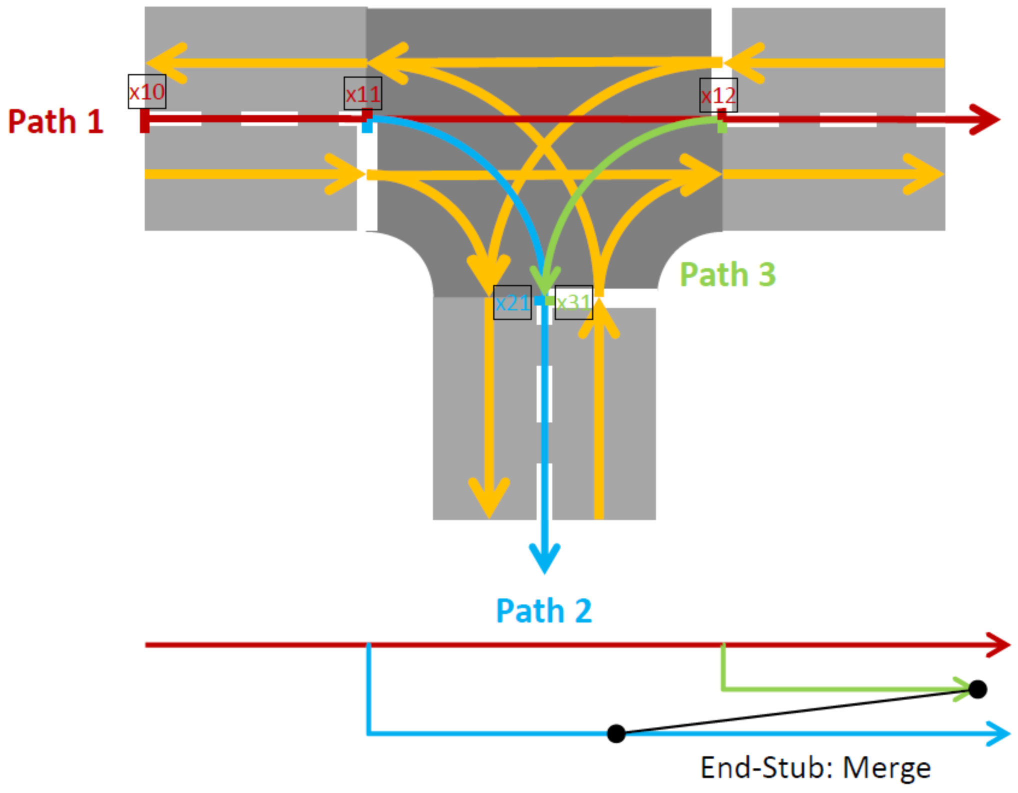


Figure 42 T intersection use case

Level 1: Single path:

- Lane description:
  - at path 1, offset x10
  - at path 1, offset x11
  - at path 1, offset x12
- Lane markings:
  - at path 1, offset x10
  - at path 1, offset x11
  - at path 1, offset x12
- Lane connectivity:
  - at path 1, offset x11
  - at path 1, offset x12

- Lane geometry:
  - at path 1, offset x10
  - at path 1, offset x11
  - at path 1, offset x12

Level 2: Additionally with drivable side paths:

- Node:
  - at path 1, offset x11:
    - > arm starting path 2
- Lane description:
  - at path 2, offset 0
  - at path 2, offset x21
- Lane markings:
  - at path 2, offset 0
  - at path 2, offset x21
- Lane connectivity:
  - at path2, offset 0
  - at path 2, offset x21
- Lane geometry:
  - at path2, offset 0
  - at path2, offset x21

Level 3: Additionally with non-drivable intersection paths:

- Node:
  - at path 2, offset x12:
    - > arm starting path 3
- Node:
  - at path 2, offset x21:
    - > FORECAST: arm ending path 3 with offset x31 (not supported in the current version)
- Lane description:
  - at path 3, offset 0
- Lane markings:
  - at path 3, offset 0
- Lane connectivity:
  - at path 3, offset 0
  - at path 3, offset x31
- Lane geometry:
  - at path 3, offset 0

Level 4: Additional collision point information:

- Collision points:
  - path 1, offset x11a, lane 1 with path 3, offset x30b, lane 1
  - path 1, offset x11b, lane 1 with path 2, offset x20b, lane 2
  - path 2, offset x20a, lane 2 with path 3, offset x30a, lane 1

9 References

[1] ADASIS Forum, "ADASIS Forum," [Online].

[2] The ADASIS Forum, "ADASIS v3 Protocol Reference".