ADASIS v2 Protocol

Version 2.0.1.0

February 2011
Proprietary and Confidential

Document control sheet

ERTICO project title: Advanced Driver Assistance Systems Interface Specifications

Document title: ADASIS v2 Protocol

Deliverable number: D2.2

Document reference:

Electronic reference: 200v2.0.1-D2.2-ADASIS v2 Specification.0.doc

Main author(s) or editor(s): Sinisa Durekovic (NAVTEQ)

Other author(s): Alexander Bracht (Daimler), Bernd Raichle (Daimler), Manuel Rauch (Continental), Julian Requejo (Ford), Dmitri Toropov (Ford), Axel Varchmin (Bosch)

Contributing author(s): Dirk Balzer (OPEL/GM), Martin Griesbeck (Continental), Jan Löwenau (BMW), Sabine Marwitz (TeleAtlas), Michel Mittaz (Bosch/Blaupunkt), Christian Ress (Ford), Nic Smith (NAVTEQ), Stephen T´Siobbel (TeleAtlas), Michael Wagner (OPEL/GM)

Special thanks to the peer reviewers: Klaus Mezger (Daimler), André Rossbach (Elektrobit), Thilo Schaper (Carmeq), Ingolf Schönherr (Bosch), Stefan Hillenbrand (Bosch), Anja Wahl (Bosch), Thomas App (Bosch), Johannes Stille (Navteq)

Document history:

Version	Status	Date	Main author	Organization	Summary of changes
2.0.0.A	Release Candidate	Feb 15th 2010			
2.0.0.0	Release	April 13th, 2010			• META-DATA message revised • CAN layouts revised • Document structure modified • Number of smaller changes in text
2.0.0.1	Release	August 12th, 2010			• Changed ERTICO logo
2.0.1.A	Proposal	Dec 1st, 2010	S. Durekovic	NAVTEQ	• Added proposal for ALTITUDE profile. • Added proposal for Bezier PROFILE (section 8.4). • Added proposal for Permanent Link Id PROFILE (section 8.5).
2.01.B	Release Candidate	Dec. 17th, 2010	S. Durekovic	NAVTEQ	• Modified proposal for ALTITUDE profile. • Modified proposal for Bezier PROFILE (section 8.4). • Modified proposal for Permanent Link Id PROFILE (section 8.5). • Minor re-formattings.
2.0.1.C	Release Candidate	Dec. 22th, 2010	A. Bracht	Daimler	• Updated color coding in table 30, “Curvature Coding” in section 9.1. Color coding in line 831 does now follow the regular scheme. • Reformatted inequations of curvature decoding in section 9.1.2. Mathematical meaning did not change, but reformatting shows better the regular scheme with powers of two. • Corrected typo in Table 14 “Stub Message”. Turn Angle value 126 was defined twice. • Limited allowed number range of Altitude profile (Table 22) to -1000m to +10.000m • Minor text changes in section 8.4 and section 8.5 • Corrected typo in right boundary value “value 1” in Table 26 All accepted change requests from “ADASIS Protocol Specification Change Request List v6” are now specified.

2.0.1.0	Release	Feb 23rd, 2011	A. Bracht	Daimler	• Released all changes since version 2.0.0.1 • The version 2.0.1.0 is completely compatible to 2.0.0.0. All changes were additions of profile types or typo corrections.

Review / Circulation:

Recipient	Date
ADASIS forum members v0.60	July 15, 2009
ADASIS forum members v2.0.0.0	April 27, 2010
ADASIS forum members v2.0.1.0	Feb 23, 2011

Abstract:

The ADASIS forum seeks to standardize the interface to predict the road geometry with its related attributes ahead of a vehicle based on the vehicle's position and a digital map, so called *ADAS Horizon.*

Table of Contents

List of Figures

List of Tables

Glossary of Terms:

ADAS	Advanced Driver Assistance Systems
ADASIS	ADAS Interface Specification
Av2	Version 2 of the ADASIS protocol definition (to distinguish between ADASIS version 1 that has been developed and tested in the Prevent Maps&ADAS project)
CAN (Controller Area Network)	A message based protocol used in automotive applications to allow devices to communicate
ADAS Horizon	Extract of the Map Database in front of the vehicle's current position
Horizon Provider (Av2HP)	The software component that generates the Electronic Horizon for transmission to other applications
Horizon Reconstructor (Av2HR)	Implementation of the Av2 Programming Interface that rebuilds the Horizon produced by the Av2HP
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
Stub	The attributes defining the start of a new path
Profile	The characteristics of a path at a specific distance along the associated 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
Main Path (Most Probable Path, MPP)	The most likely route of the vehicle on the road, represented as a single Path
Sub Path	A path (other than the MPP) that the vehicle may potentially follow
First level sub path	Sub path that is directly attached to the main path
Second level sub path	Sub path that is directly attached to a first level sub path
Parent Path	Path to which a sub path is attached

Transmitted Path	Path that has been actually transmitted from the Provider to the Client via the CAN network
Path Index	Path identification number as used in the ADASISv2 CAN messages
Path Identifier	Internal path identification number of the horizon provider or horizon reconstructor that may have more values than the Path Index
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 using a few control points applied to a polynomial function
Clothoid	A curve whose curvature increases linearly with distance and describes the optimal form of a curve in the road network
Message Type	Type of a CAN message in the scope of the ADASISv2 protocol
Multiplexing	Usage of a multiplexor ID (= message type) to define different CAN message layouts under one CAN ID

1 Introduction

The ADASIS forum [1] and the Prevent/Maps&ADAS project [2][5][6][7] have developed a specification to describe the road geometry with its related attributes ahead of a vehicle based on the vehicle's position and a digital map (known as the *ADAS Horizon)*. This specification ("ADASIS v1") has been prototypically implemented on a CAN bus by several partners and the performance and interoperability of the implementations have been successfully tested. The ADASIS concept has proven to be technically feasible.

However, the ADASISv1 interface has turned out to be rather complex to implement in the OEM production environment. In particular, the client ADAS applications needed to implement sophisticated horizon reconstruction software, which threatened the success of the standardization approach.

This document specifies a simplified ADASIS interface ("ADASIS v2") that makes the interpretation and reconstruction of ADAS Horizon information much easier.

In the general architecture of ADAS applications, the main entities are:

- **ADAS Horizon Provider,** which maintains the ADAS Horizon;
- **ADAS Protocol** that defines how the ADAS Horizon will be sent from the ADAS Horizon Provider to the ADAS Applications;
- **ADAS Application** is a client application that receives the ADAS Protocol messages then reconstructs and uses the ADAS Horizon;
    - **ADAS Reconstructor** is a common component of ADAS Applications that is built in accordance with this general architecture. The task of the ADAS Reconstructor is to receive, parse and interpret ADAS Protocol messages, and, in effect, reconstruct a copy of the ADAS Horizon on the client side.

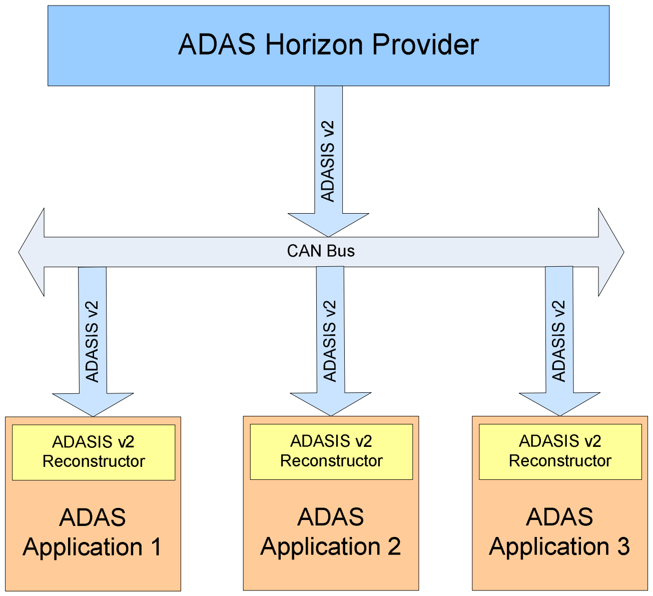


Figure 1: ADASIS v2 System Architecture

In this document we will refer to above entities as ***ADASIS v2 Horizon Provider*** (***Av2HP***), ***ADASIS v2 Protocol*** (***Av2***) and ***ADASIS v2 Horizon Reconstructor*** (***Av2HR***). In addition, the ADASIS v2 Protocol assumes a CAN bus communication channel between the ADASIS v2 Horizon Provider and the ADAS Applications.

2 *Versioning*

2.1 Protocol versioning

The ADASIS v2 Protocol version is described with three numbers separated by points: **X.Y.Z**.

- **X** is the major version number. It is number 0, 2 or 3. Pre-releases of ADASIS v2 protocol will contain a major version number of 0; the first official release will have the major version number 2; the next version of protocol will be designated by the major version number 3.
- **Y** is the minor version number in the range 0-15. The minor version number will be increased each time the syntax of the protocol is changed, such as: new or removed messages, new or removed message fields, changed number of bits of a signal etc.
- **Z** is the minor sub-version number in the range 0-7. The minor sub-version number is increased each time the semantics of the protocol are modified – for instance, interpretation of field values or similar.

The number ranges are defined according to the version signals in the META-DATA message.

2.2 API versioning

API version numbers follow the same convention as the protocol. In general, versioning of the API is independent of the versioning of the protocol.

2.3 Versioning for this document

Versions of this document, *ADASIS v2 Protocol*, are designated by **X.Y.Z.W**. The first three fields are the same as the corresponding protocol version. The last field, **W**, indicates changes in the document that are not related to changes in the protocol – revision of the text, changed or extended descriptions (but when syntax and semantics are kept intact), etc.

Working versions are labelled A, B, C. When **W** is a number 0 to 999, the specification document is approved for release.

The current version of this document is Version 2.0.1.0.

Because of naming conventions of the ERTICO documents, the name of this file will be in the form:

`200v`***`X.Y.Z`***`-D2.2-ADASIS v2 Specification.`**`W`**`.doc.`

For instance,

```
200v0.7.3-D2.2-ADASIS v2 Specification.C.doc
```

contains the third working version (**C**) of the **7**th revision of the protocol syntax and the **3**rd modifications of the semantics of the draft protocol specification.

Similarly,

```
200v0.7.3-D2.2-ADASIS v2 Specification.20.doc
```

refers to the same protocol syntax and semantics, but there were a number (20) of changes in the document such as better descriptions, new pictures etc.

The 200 at the beginning of the filename indicates that this document is the 200th of the ADASIS forum. The 200 is not version information. The "D2.2" means that this document is the **2**nd delivery document of the working group **2** in the ADASIS forum.

Please note that as auxiliary document to this one can find the file Vector CANdb++ that follows similar convention

`200v`***`X.Y.Z`***`-D2.2-ADASIS v2 CAN Database.`**`W`**`.dbc.`

3 ADASIS v2 Concept

In the digital map database, the road network (Figure 2) is represented as a collection of *links* (or *segments*) and *nodes* that define connectivity between links (Figure 3).

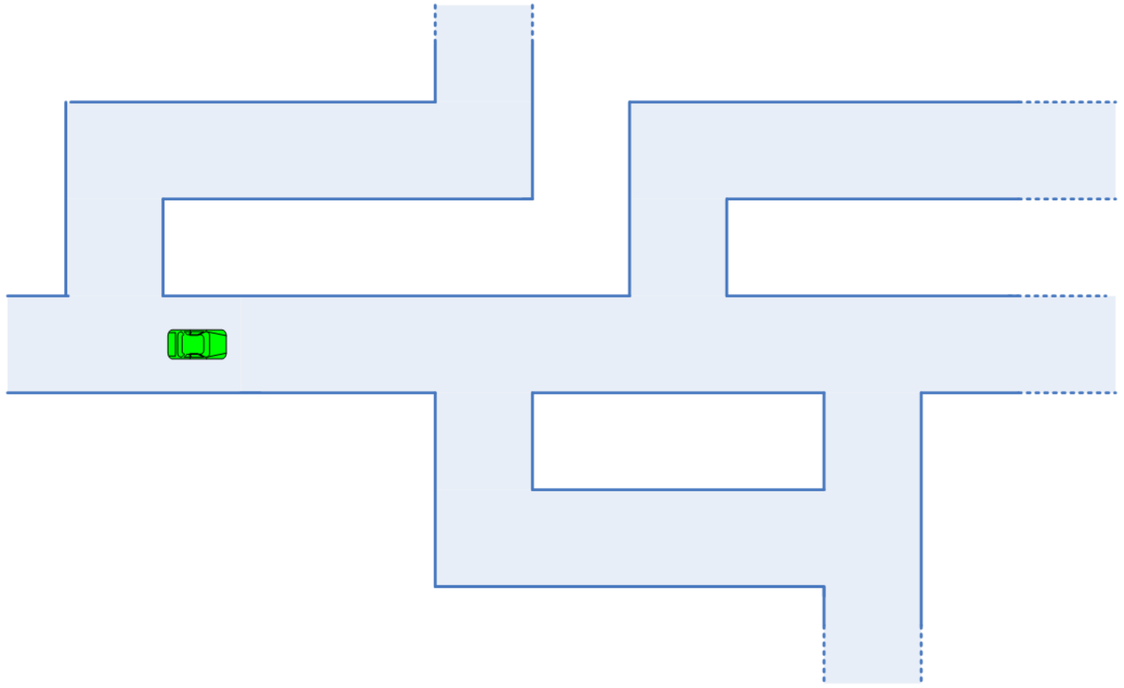


Figure 2: Road network

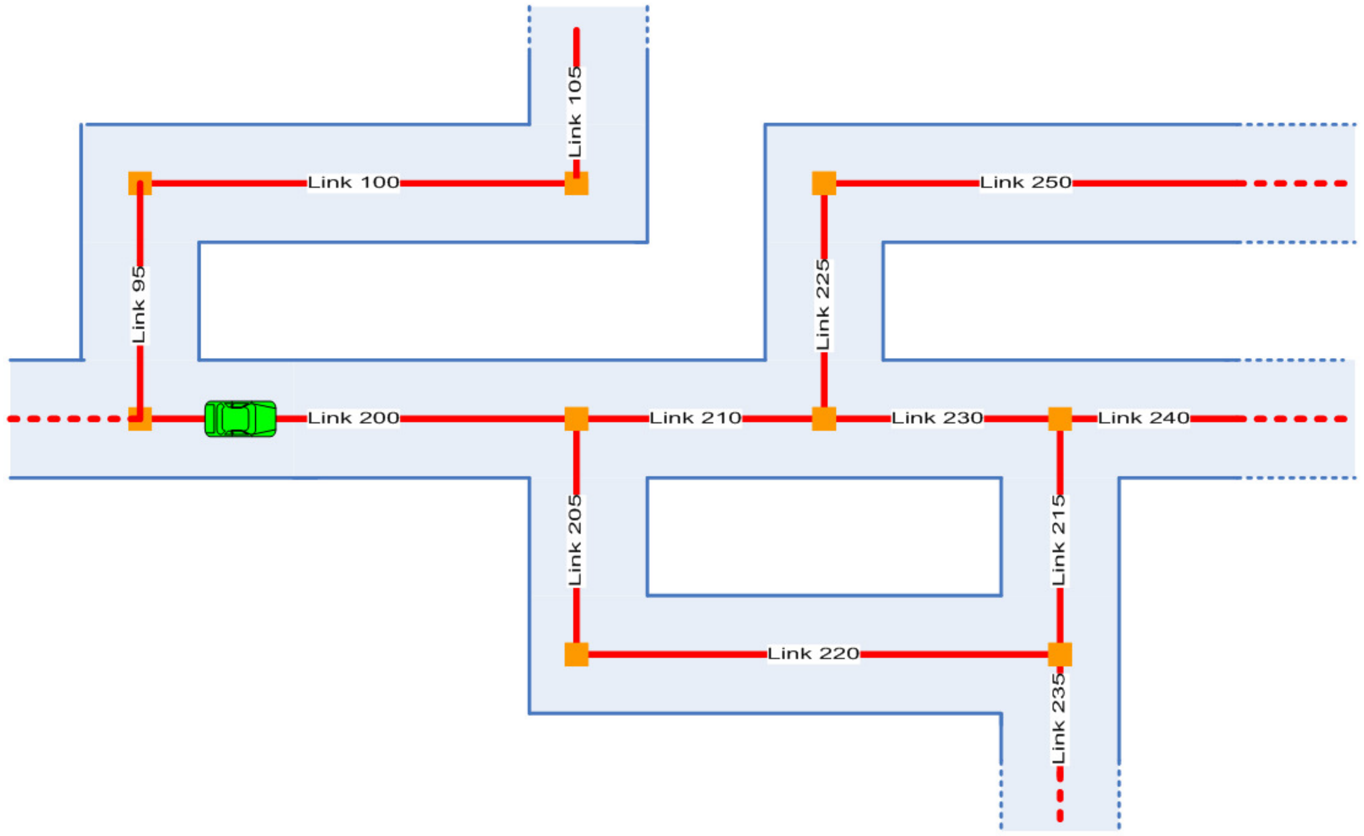


Figure 3: Digital Map Database

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

The ADAS Horizon (*Electronic Horizon, eHorizon…*) is the part of the digital map that contains only those roads in front of the vehicle (Figure 4).

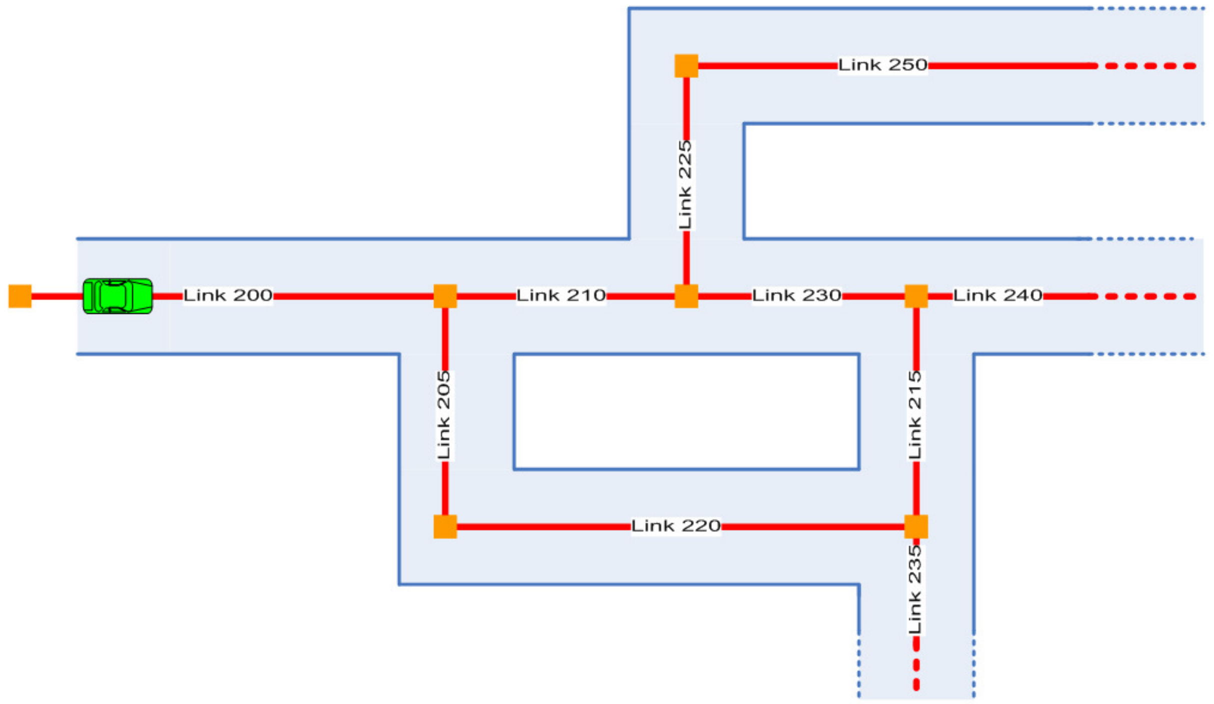


Figure 4: ADAS Horizon

Comparing a simple extract of roads around the vehicle in Figure 3 and the ADAS Horizon in Figure 4, one can see that links 95, 100 and 105, being not important for majority of ADAS applications, are not in the ADAS Horizon. In addition, not all link attributes available in the digital map need to be present on links of the ADAS Horizon. For instance, street names or house number ranges will rarely be needed by ADAS applications.

In other words, the ADAS Horizon provides to ADAS application an optimized view of the environment, allowing 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. One such implementation is described in [14].

The ADAS Horizon can be represented as a *Network of (Digital Map) Links*, as shown in Figure 4. For the ADAS application, this view is rather complex because it must analyze the network in order to get information about entities 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.

It is more convenient for the ADAS application to deal with *paths* – trajectories the vehicle may follow in the near future. Internally, paths are built from database links and their connectivity, but each path is seen by the application as a single entity.

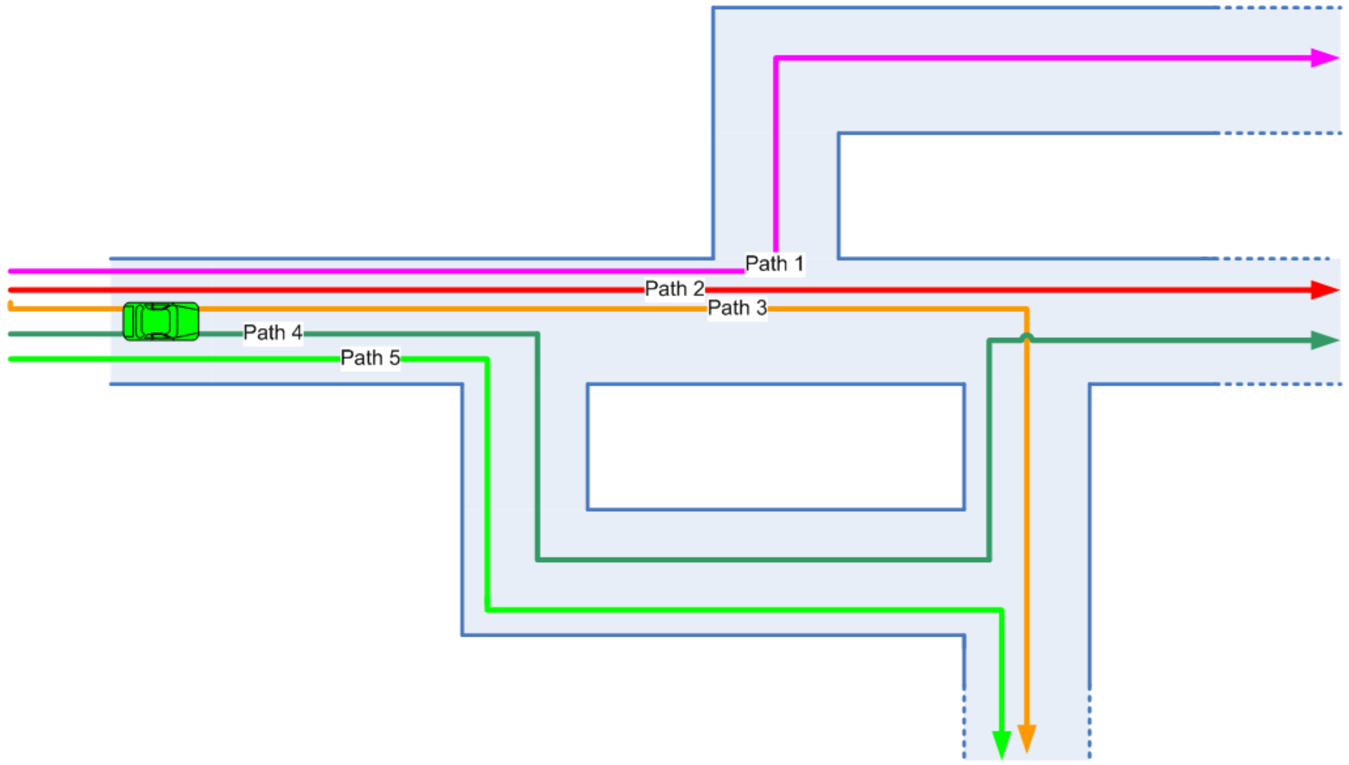


Figure 5: ADAS Horizon – (Pure) Path representation

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 5). 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 5, 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.

Optimized Path Representation in ADASIS v2 reduces, but does not fully remove, such redundancies.

3.1 ADASIS v2 or Optimized Path Representation of the ADAS Horizon

The ADASIS v2 concept describes an ADAS Horizon using *Optimized Path Representation.* This approach reduces the amount of duplicated data, but still provides most of the advantages of the path approach over a network representation of the ADAS Horizon (Figure 6).

As a minimum, in Optimized Path Representation just one path needs to be present, let us call this the *Main Path*. The defining characteristic of the Main Path is that the current vehicle position is located on it. In Figure 6, Path 2 is the Main Path.

The vehicle may turn from the Main Path to *First-Level Sub-Paths*. In the above example Paths 1, 3 and 4 are first-level sub-paths.

From first-level sub-paths there are possible turns to *Second-Level Sub-Paths* (Path 5, for instance, from Path 4), etc.

The ADAS Horizon’s construction algorithm should choose the Main 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 on the Most Probable (Main) path.

In distributed environments, with one Av2HP and one or more Av2 applications, question how many level of sub-paths shall be sent depends of the application tolerance to lack of horizon. If few seconds of non-functioning are acceptable, only main path can be sent over communication channel. After vehicle has left most probable path, some time will be necessary to send new one.

If application must have horizon data all the time, full horizon shall be transferred. This ensures that, even if vehicle leaves one path, it will automatically be positioned on another path that is already available at the client side.

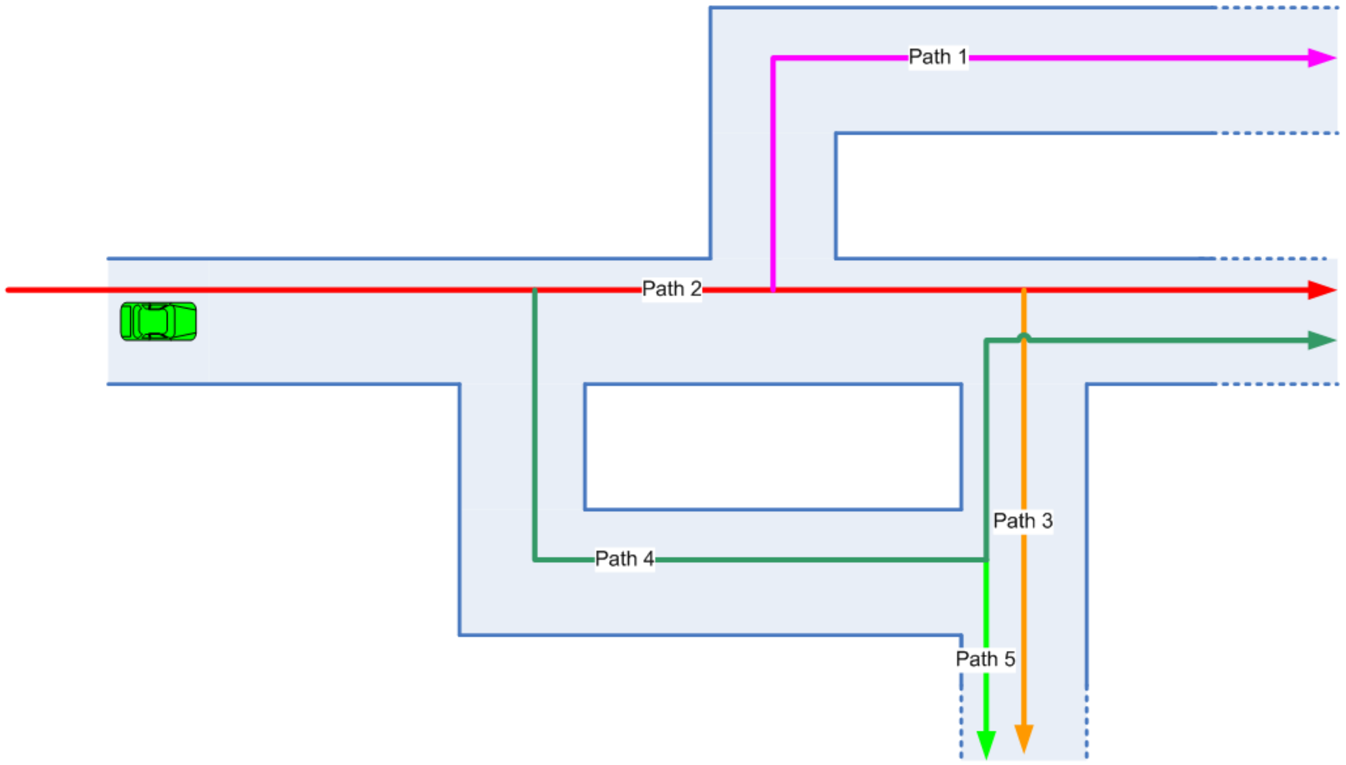


Figure 6: ADASIS v2 Horizon - Optimized Path Representation

Since ADASIS v2 views the ADAS Horizon using Optimized Path Representation, let us call this representation the *ADASIS v2 Horizon*.

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

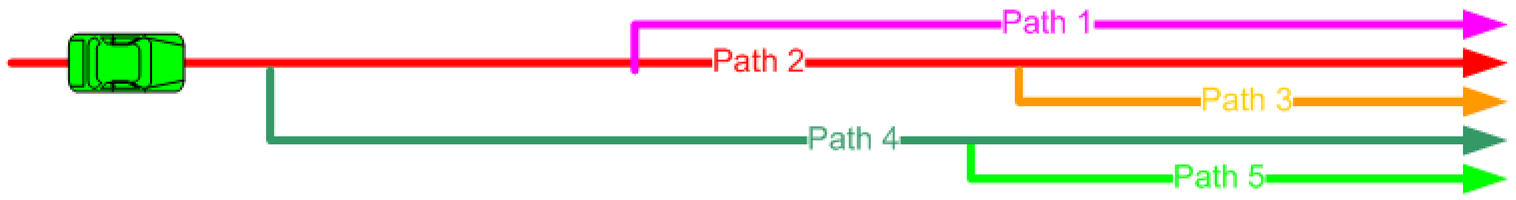


Figure 7: ADASIS v2 Horizon - Application View

From the application viewpoint (Figure 7), the vehicle is located on one (main) path; the data of interest are either on the same path or on one of sub-paths ahead of the vehicle.

3.2 ADASIS v2 Building Blocks

As we have seen, the ADASIS v2 Horizon consists of *paths*. Each path is uniquely identified by a *path identifier.*

It will be shown that the relationships between paths are defined by entities called *stubs*. Each stub marks the start of a (sub-) path and it is located on another (parent) path. We will also introduce the concept of *profiles*, which are the characteristics of paths.

The position of Stubs and Profiles on the ADASIS v2 Horizon are defined by a path identifier and an *offset* along that path. All offsets are defined as the distance between an entity and the start of a path.

The current *Vehicle status* is described in terms of its position in respect to the Av2 Horizon (again path identifier and offset) and other characteristics such as speed and heading.

3.3 Possible Configurations of the ADASIS v2 Horizon Provider

In general, ADASIS v2 Horizon Providers should initially consider and calculate all possible paths.

However, the client ADAS application may need just a subset of those paths. Most applications will actually only need the main path. Therefore, in many installations it will be enough if the ADASIS v2 Horizon Provider (Av2HP) sends only the main path to the client (see Figure 8).

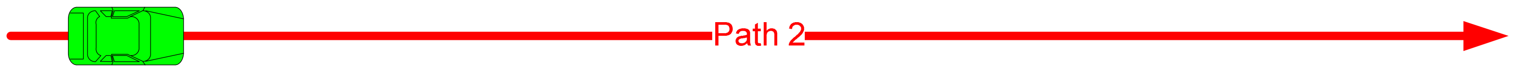


Figure 8: Main Path only

If only the main path has been provided, but the vehicle leaves it, the application will be “blind” for a moment until new information for the new path is available.

If the ADAS application also needs information for sub-paths, the ADASIS v2 Horizon Provider may offer extended configuration levels, also transmitting preview information reaching into sub path structures. This is done by using "stubs" which indicate the start of a new path attached to a parent path, and which contain basic information about the attached road (e.g. turn angle, road class). The detailed definition of "stubs" is described in chapter 4.6.

Starting from Figure 8, a next configuration level for the Horizon Provider would be the Main Path with attached stubs, which contain basic intersection information (see Figure 9).

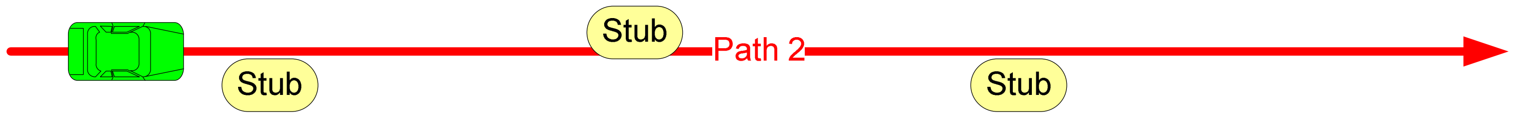


Figure 9: Main Path with Stubs

If the application is sensitive to a short "blindness" when the vehicle leaves the main path then the Horizon Provider must also preventively transmit information about upcoming sub paths. With the availability of sub paths, the client will receive the information when the vehicle position has, for example changed from Path 2 to Path 1, no gap will occur in the available track preview data on the client side.

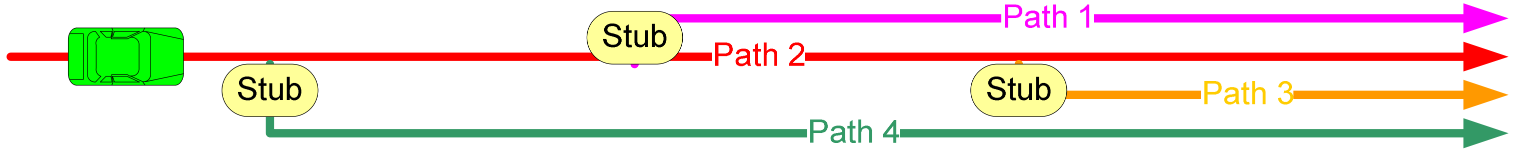


Figure 10: Main Path with first level Sub-paths

If the client needs even more information and has enough memory to store it, the ADASIS v2 Horizon Provider can be configured to transmit the full available map information with higher-level sub paths as shown in Figure 11. In that case, the client will never be in a "blind" state and will always have immediate map information available when the vehicle leaves a path.

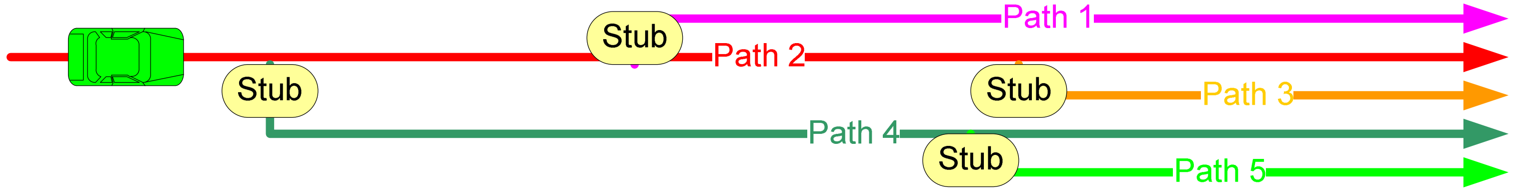


Figure 11: Full Horizon

The configuration of the ADASIS v2 Horizon Provider and the completeness of the transmitted horizon is installation specific and not defined in this document. Each vehicle installation may have a different configuration, depending on the requirements of the applications, the available bus bandwidth, and the storage capabilities of the application control units.

It can be assumed that there are simple applications needing only the main path transmission, as well as more complex applications needing the full horizon in one vehicle set-up. In this case, the ADASIS v2 Horizon Provider would provide the full horizon, but the simple application would be allowed to delete at the CAN receiver interface all information it cannot store or does not need. The ADASIS v2 protocol is designed in a way that simple applications are not forced to implement more ADASIS v2 functionality than they absolutely need and that more complex applications can use additional information layers (see also section 8.1).

3.4 Paths and Offsets

3.4.1 *Definition of Offsets*

As already described, each path is uniquely identified by some number (*Path Identifier),* the position of map entities and the vehicle position is defined by the path identifier and the distance from the start of the path. This along-path distance is called *Offset*.

The start of each path is defined to be at offset zero.

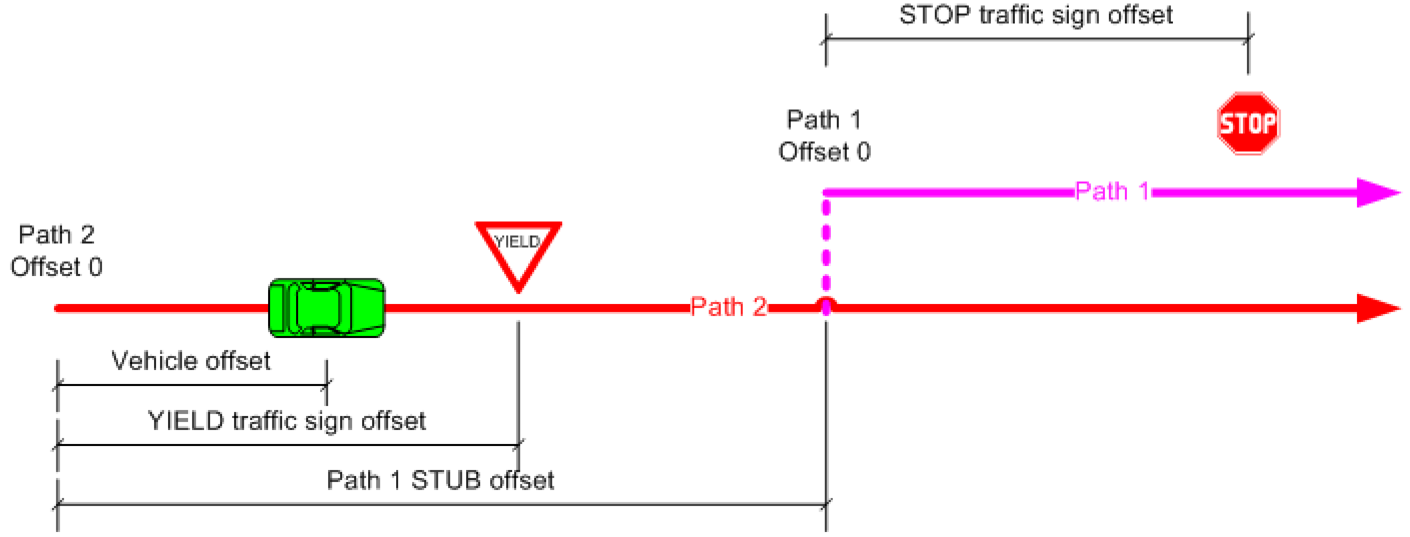


Figure 12: Positioning of entities on ADASIS v2 Horizon

At intersections encountered on the main path, sub-paths may branch off. These entries into branches are called *stubs*. Like other map entities, the location of *stubs* along the ADASIS v2 Horizon is described by specifying the (parent) path identifier and offset. Physical crossings on the path can be described as locations with one or more stubs – one for each alternate road at the crossing.

From an abstract perspective, both the ADASIS v2 Horizon Provider and the ADASIS v2 Horizon Reconstructor/Client do not need to have any limitation of the number of paths present in each moment and of the length of each path. A path has a defined start point (offset 0), but potentially no end at all. Therefore, the offset can be any large positive number.

As we will see later, the ADASIS v2 Protocol defines only 6 bits for the identification of the path that is referenced by each CAN message. For the offset, just 13 bits are available. After removing special values, an ADASIS v2 message accommodates 56 numbers that can be used for path identification and offsets between 0 and 8190 meters inclusive.

For many ADAS applications, the limitation to maximum 56 paths and a maximum length of approximately 8 kilometres will not raise any problems. Many ADAS applications such as map-based Adaptive Cruise Control (ACC) or adaptive curve lighting will require only a short horizon length and very few paths.

3.4.2 Cyclic Offset Value

The ADASISv2 CAN message definition allows only a limited number range to encode the offset value (13 bits). However, if the message OFFSET field is used to store cyclic values and if the order of the messages is taken into account, practically unlimited path lengths can be achieved.

The following figures show an example for the "main path only" configuration with a maximum length of transmitted path of 1000 m and an offset wrap (= bit overflow) from 8190 to 0 at the end of the maximum offset range.

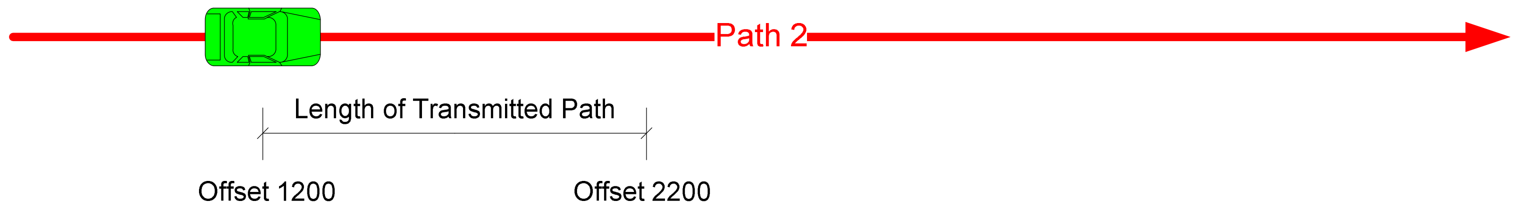


Figure 13: Short ADAS Horizon

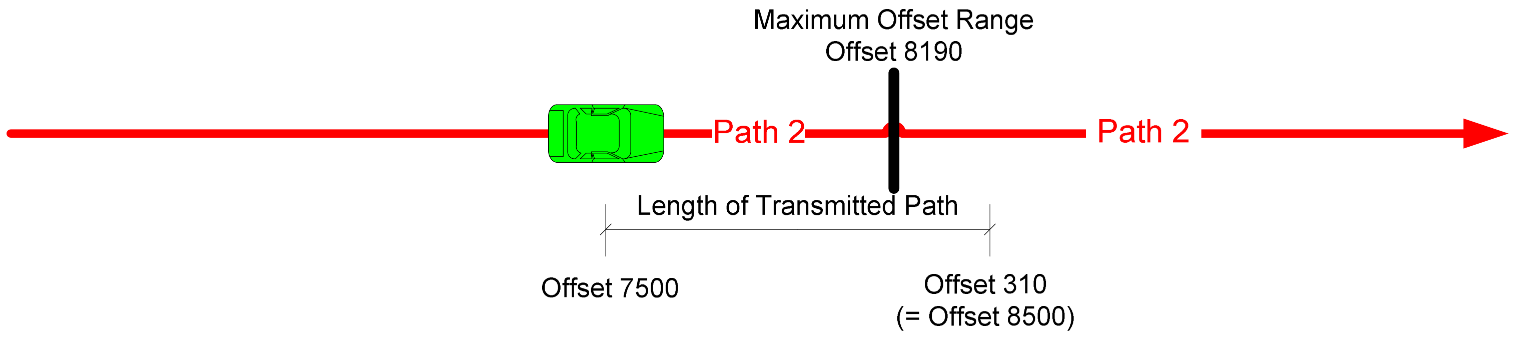


Figure 14: Limited Cyclic Offset

The whole ADASIS v2 concept is event-based, which means that the Av2HP will not send out new horizon messages until there is a change of attribute or a new event to be announced. As the current length of the transmitted path is marked by the outmost event location, this length can be less than the maximum length of transmitted path. In this case, the client can assume that no map attributes have changed up to the defined maximum length of transmitted path.

3.4.3 Path Length Limits

The ADASIS concept uses different range values to define the length of the ADAS horizon. These are defined in this chapter.

Maximum Offset Range:

Allowed number range to define an offset value in each CAN message: 0 – 8190 (13 bit).

Maximum Length of Transmitted Path:

This value configures the ADAS horizon provider and defines, that the provider shall never transmit information farther away from the current vehicle position than "maximum length of transmitted path".
The "maximum length of transmitted path" may be smaller than the "maximum offset range", but the horizon provider shall still use the complete available number range for the cyclic offset value.

Current Length of Transmitted Path:

This value is dynamic and changes during driving. The whole ADASIS v2 concept is event-based, which means that if the map attributes do not change, the Av2HP will not send out new horizon messages. Consequently, the "current length of the transmitted path" is not guaranteed and can vary. In this case, the client can assume that no map attributes have changed up to the defined "maximum length of transmitted path".

The "current length of transmitted path" depends on the message type and on the profile type. It is defined by the distance of the current vehicle position to the farthest horizon message offset.

Example:

- Figure 15 shows an example of the different length values.
- The current vehicle position is at offset value 1000m
- The "maximum offset range" is limited to 8190
- The "maximum length of transmitted path" is configured to 1200m
- The "current length of transmitted path" for the short profile slope is 1000m, because the provider has sent out a new slope information 0% at offset value 2000
- The "current length of transmitted path" for the segment message (containing the road class) is only 500m, because the farthest road class information is at offset 1500 and the next one at offset 2500 is too far away and has not yet been transmitted.

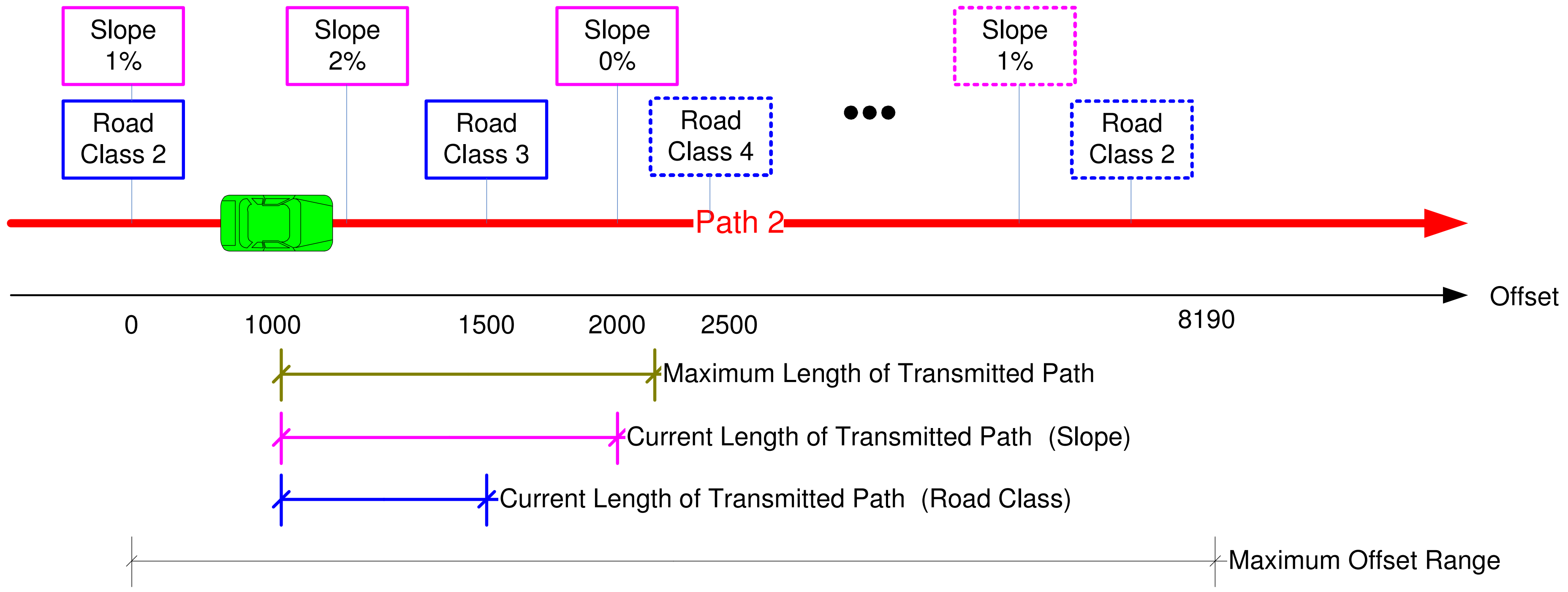


Figure 15: Definition of Path Lengths

As the road class example in Figure 15 shows, the "current length of transmitted path" may become rather short if there is no attribute change to transmit. To avoid this, the horizon provider should send intermediate messages after a certain distance, that repeat the information but give more reliability for the ADAS client (see also ADAS v2 Horizon Provider (Av2HP), Av2HP Recommendation #6).

3.4.4 Unlimited Horizon Length

There are two options how Av2HP can deal with offset range limitation. The algorithms are described in more detail in section 5.7.

- **Unlimited paths / Cyclic offset**
  The Av2HP can send out map entities with offset positions beyond the maximum offset range. In that case, the Av2 client needs to use the order of the incoming messages to distinguish between the offset values on different "alias levels" of the offset value.

- **Limited paths / Non-cyclic offset**
  When an offset reaches the maximum value (8190) the Av2HP can start a completely new path with a new Path Identifier. With this method, new paths can be appended to the main path (or any path) to provide very long horizon information.
  This method is not compatible with cyclic offset approach. A simple application in the same vehicle that would need only the main path like in Figure 13 would be forced to interpret the path/sub-path concept with the parent/child connection.
  Main disadvantage of this approach is fact that application needs to construct single logical path from several Av2 paths by itself.
  A second disadvantage of this method is, that a complete tree of paths will be set up and that many Path Identifiers may be needed. To solve this problem, path identifiers could be used multiple times within the valid horizon and the client would need to use the receiving order of the paths to distinguish.

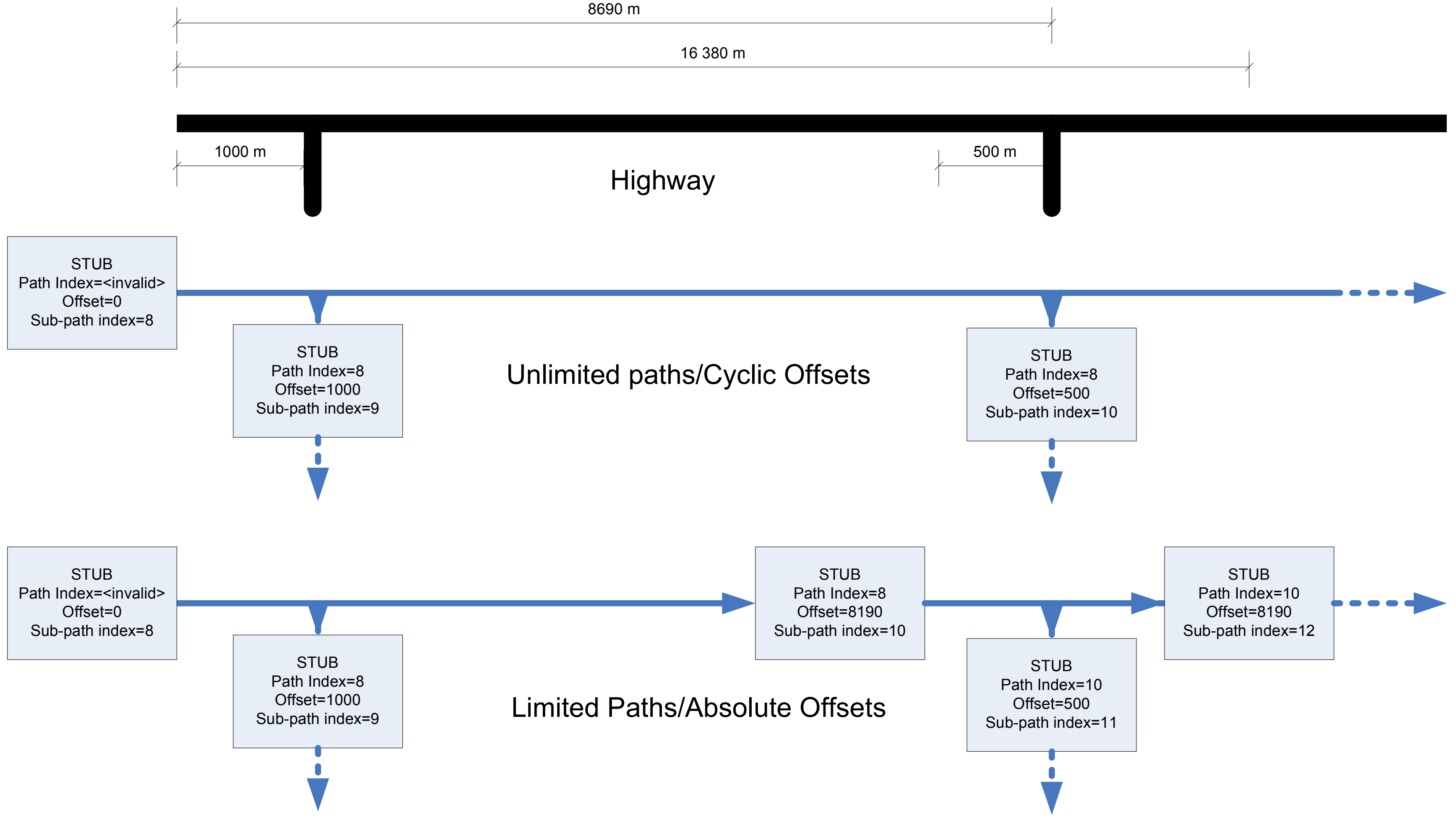


Figure 16: Highway scenario - Unlimited cyclic vs. Limited non-cyclic Path representation

3.5 Path Profiles

Path Profile is a property that has a value for any location along a path (e.g., curvature, form of way, number of lanes, speed limit, horizontal geometry). This value may also be <undefined>.

A *Path Profile* is made up of a quantity of "**Profile Spots**" and the specification of a **Profile Interpolation Type**.

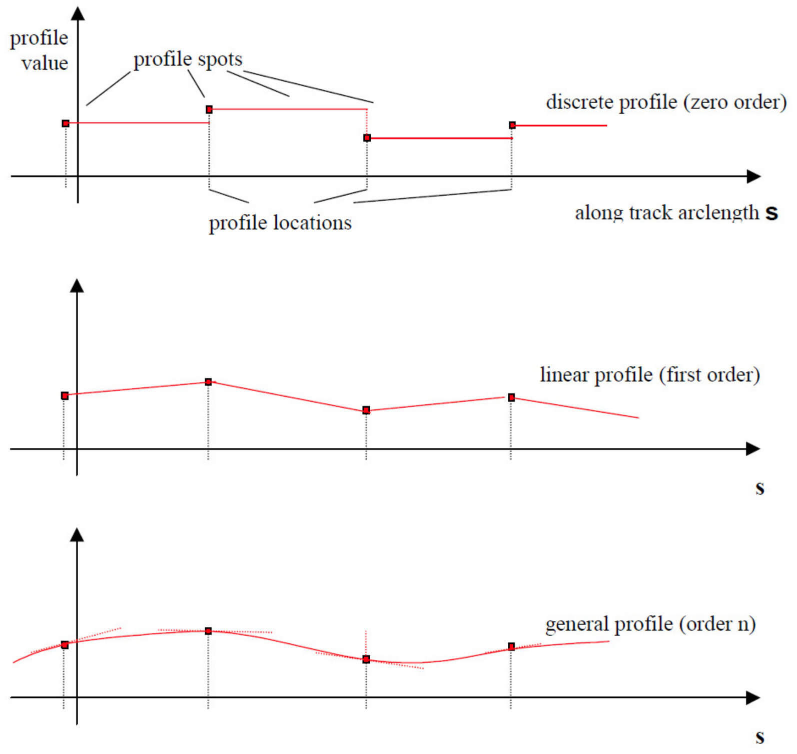

profile value
profile spots
discrete profile (zero order)
profile locations
along track arclength s
linear profile (first order)
s
general profile (order n)
s

Figure 17 Profiles

As there are a variety of ways to describe a profile as a function of offset along the path, the *Profile Interpolation Type* specifies how the profile value is to be calculated from the *Profile Spots* at intermediate locations using some interpolation scheme (e.g., discrete, step-profile, linear or higher order interpolation …).

Figure 17depicts the key definitions involved in path profile representations, together with visualized examples for different *Profile Interpolation Types*.

- ***Profile Type*** is the property that the profile shall represent.
- ***Profile Interpolation Type*** is a code specifying how intermediate profile values are to be calculated.
  The following Profile Interpolation Types are identified as follows:
  - Discrete interpolation profile (zero order): The profile is specified by the set of locations of attribute changes together with the new value. Each value is valid until next location.
  - Linear interpolation profile (first order): The profile is specified by a set of locations together with the attribute value at the location, where the value must be linearly interpolated between the given locations. (Note that the polygonal 2D representation of the map geometry is a first order profile, since the geometry between the shape points is calculated by linear interpolation of latitude/longitude values).
  - Higher order interpolation profile (quadratic / cubic polynomial, spline ...): The profile is 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 must be specified. (Note that this may be done by using the extended set of coordinates plus attributes from the two neighbouring points and/or additionally use values from second-degree neighbours; alternatively, *control points* can define additional parameters used in interpolation (see also section 5.8).
  - Attachment (singularity spots): The attachment is specified by the set of locations of entities with an attribute value at each location. This is in fact a degenerated profile, since there are no entities and therefore no attribute values between the given location spots. A typical attachment is a Traffic Sign.

  In ADASIS v2, *Profile Type* implies *Profile Interpolation Type*. For instance, a step-interpolated Slope profile is of a different type than a linear-interpolated Slope profile.
- ***Profile Offset*** is a position along a path, defined by its distance from the path origin, measured as the arc or poly-line length along the path.

- ***Profile Spot*** is the numeric description value of a property at a Path Offset (note that this may be a single value p_k, representing the property value itself at *Path Location k*). Depending on the interpolation type, a *Profile Spot* may also be defined as a vector p_k containing several values (including for example first or second order derivatives or longitude + latitude coordinates) necessary for smooth curve representation (e.g., for polynomial or spline interpolations). In that case, profile *control points* are used to accommodate additional parameters (see also section 5.8).

For more details, see also [3].

4 Messages

ADASIS v2 assumes one-way communication between an ADASIS v2 Horizon Provider and ADASIS v2 Horizon Clients. Six types of messages are defined in the ADASIS v2.

In this section, those messages are ordered by priority. Besides a list and descriptions of message fields, the CAN layout for each message is also defined. Five bits are common to every ADASIS v2 message:

1. The three bits define message type. This field is critical in multiplexing setups (Table 2), in a non-multiplexing CAN environment message type can be deduced from the CAN identifier of message (Table 3) and this field is redundant. TYPE field always occupies bits 7-5 of byte 0 of each CAN frame.
2. Two bits of each message are a cyclic message counter for each specific message type (and each profile type). It will be used to detect missing frames at the client side.

The following message types are defined:

- POSITION message specifies the current position(s) of the vehicle.
- STUB message indicates the start of a new path that has origin at an existing one.
- SEGMENT message specifies the most important attributes of a part of the path.
- PROFILE SHORT message describes attribute of the path whose value can be expressed in 10 bits.
- PROFILE LONG message describes attribute of the path whose value can be expressed in 32 bits.
- META-DATA message contains utility data.

The value definition of each signal in the messages includes two additional special values "unknown" and "signal not available". The "unknown" value should be used by the ADAS horizon provider, if a certain information is not known for one road (e.g. unknown in the map data if a certain road is a tunnel or not). The "signal not available" value (abbreviation "N/A") should be used, if a certain data attribute is completely not available or implemented (e.g. a certain map does not have tunnel information at all or a certain software release has the tunnel information not yet implemented).

4.1 Message Types and CAN frame Multiplexing

In each message, the 3-bit field ***Message Type*** determines semantics of rest of the message.

Message Type	***Message***
0	Reserved
1	POSITION
2	SEGMENT
3	STUB
4	PROFILE SHORT
5	PROFILE LONG
6	META-DATA
7	Reserved

Table 1: Message Type field

Therefore, the content of the *Message Type* field determines what fields are present in each CAN frame used in the protocol.

The minimum number of CAN identifiers that must be assigned to ADASIS v2 protocol is one.

For instance, the following convention can be used:

CAN Identifier	Content of Message Type field	Message
n/a	0	Reserved
100	1	POSITION
100	2	SEGMENT
100	3	STUB
100	4	PROFILE SHORT
100	5	PROFILE LONG
100	6	META-DATA
n/a	7	Reserved

Table 2: ADASIS v2 as fully multiplexed CAN protocol

If multiplexing (polymorphism) of CAN frames is not desirable, different CAN identifiers can be assigned to the different messages. However, the *Message Type* field is still filled with proper values.

CAN Identifier	***Content of Message Type field***	***Message***
n/a	0	Reserved
101	1	POSITION
102	2	SEGMENT
103	3	STUB
104	4	PROFILE SHORT
105	5	PROFILE LONG
106	6	META-DATA
n/a	7	Reserved

Table 3: ADASIS v2 as non-multiplexed CAN protocol

Most systems support single-level multiplexing in CAN traffic. If no other requirements are set, only the *Message Type* field can be used as the multiplexing field, since each message type has fixed structure.

Strictly speaking, each profile type accommodated in the PROFILE LONG and PROFILE SHORT require distinct CAN identifier, since the value fields are interpreted differently for each profile type. If, in such an environment, strict single-level multiplexing is required, one must reserve at least two CAN identifiers to accommodate ADASIS v2 protocol.

CAN Identifier	Content of Message Type field	Message	Multiplexing field
n/a	0	Reserved	
200	1	POSITION	MESSAGE TYPE
200	2	SEGMENT	MESSAGE TYPE
200	3	STUB	MESSAGE TYPE
201	4	PROFILE SHORT	MESSAGE TYPE + PROFILE TYPE
201	5	PROFILE LONG	MESSAGE TYPE + PROFILE TYPE
200	6	META-DATA	MESSAGE TYPE
n/a	7	Reserved	

Table 4: ADASIS v2 as partially multiplexed CAN protocol

As shown in Table 4, the POSITION, SEGMENT, STUB and META-DATA messages share the same CAN id and multiplexing is accomplished by looking at the *Message Type* field. PROFILE SHORT and PROFILE LONG messages share another CAN identifier. For those two message types, the multiplexing field is a combination of the *Message Type* and the Profile Type fields, which together are conveniently exactly 8 bits long.

If multiplexing (polymorphism) of CAN frames is not desirable, different CAN identifiers can be assigned to different messages. However, the *Message Type* field is still filled with proper values. If different profiles are also considered as different messages, the total number of CAN identifiers that needs to be used is 4 (POSITION + SEGMENT + STUB + META-DATA) plus the total number of PROFILEs, which may be up to 62 (31 SHORT profiles and 31 LONG profiles).

In conclusion, we see that, depending on configuration and requirements, the number of CAN message identifiers that needs to be allocated may vary between 1 (fully multiplexed protocol) to a theoretical maximum of 66 (each message type and each profile has own CAN identifier).

The developer should also take in account that the use of more than one CAN identifier may influence the order of receiving frames at the client side.

4.2 Intel CAN message layouts

CAN message layouts illustrated in following tables show the arrangement of ADASIS signals using the Motorola format (big endian). Due to its common adoption within the automotive industry, the Motorola format has been selected as the preferred CAN message format. If a particular implementation requires a layout based on the Intel format (little endian), signal start-bits must be adapted accordingly. One simple approach to accomplishing this is by shifting the byte ordering. Assuming that the least-significant bit is at position zero, this may be expressed mathematically by the following formula:

$$\mathtt{Startbit}_{\mathtt{Intel}} = \mathtt{Startbit}_{\mathtt{Motorola}} + 56 - 16 * \mathtt{floor}(\mathtt{Startbit}_{\mathtt{Motorola}}/8)$$

The system designer must realize that these two formats are not compatible with each other. In order to guarantee consistency, the provider, and all reconstructors connected to the same CAN bus must use the same format.

4.3 Alternative bus configurations

The ADASISv2 specification defines CAN messages with 8 bytes payload data. The ADASIS forum has decided to design the specification for CAN, because it has the strictest constraints compared to other current bus systems like Flexray. CAN is widely established as basic technology for data communication in the vehicle network. Due to the long change cycles in vehicle technology, CAN will still be the most important data bus in the coming years. Other bus systems like Flexray and MOST are specialized on certain applications and will not replace the complete CAN network in the vehicle.

If the ADASIS messages should be transmitted over other bus systems like Flexray, the 8 bytes payload data will not raise major problems. If the ADASIS messages should be transmitted over TCP/IP, this can be realized by introducing 8-byte binary blocks to send from the provider to the clients.

4.4 POSITION message

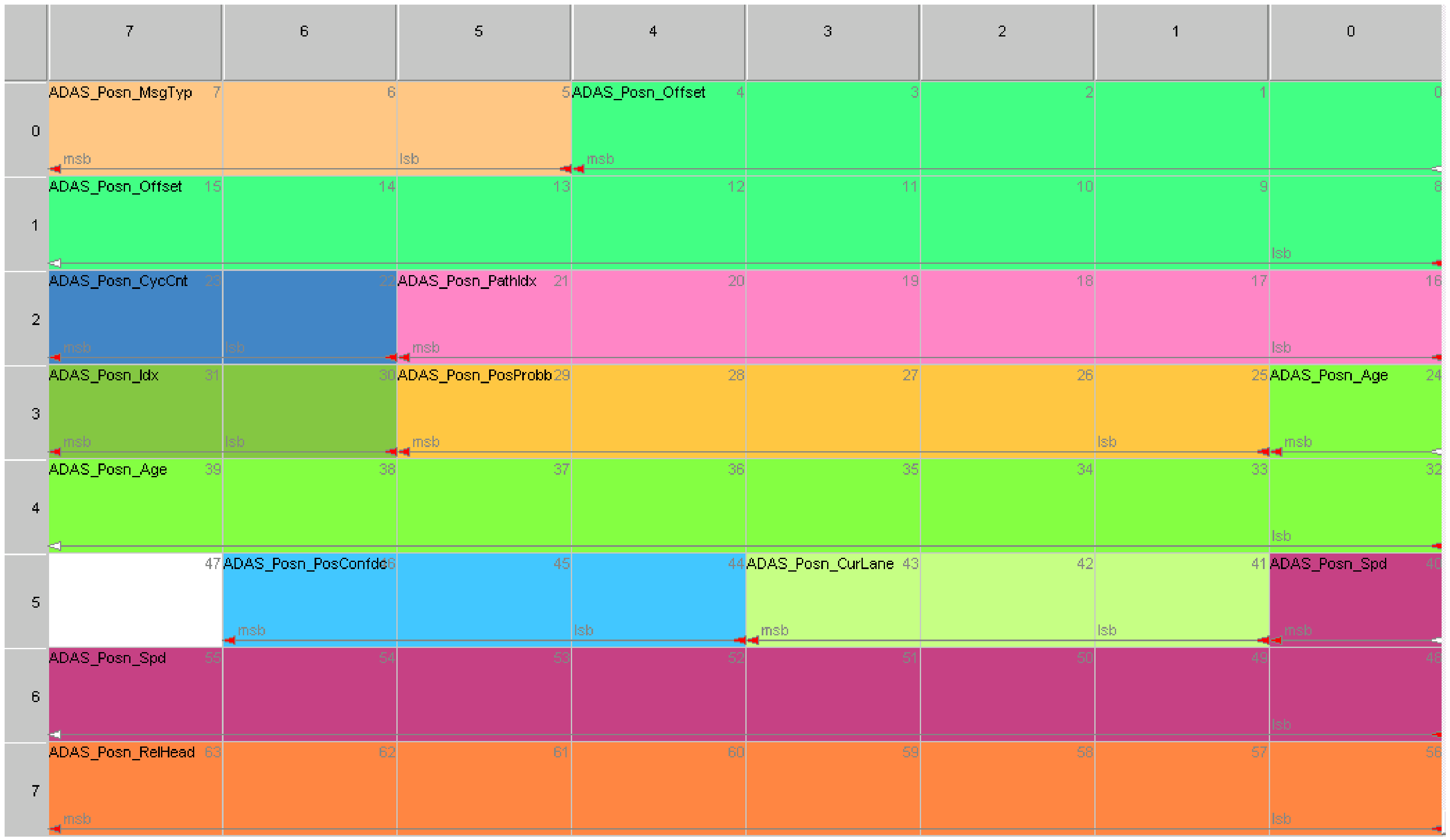
The POSITION message describes current position(s) of vehicle in relation to ADASIS v2 Horizon Paths.

Field	Length (bits)	Value Range	Units	Description
Message Type	3	1		See Table 1
Cyclic Counter	2	0..3		Cyclic counter of this message type: Reconstructor can use this value to detect missing messages
Path Index	6	0..7 8..63		Index of current path: If this number differs from one specified in previous POSITION message, vehicle left the original path; Indices 0-7 have special meaning: see Table 7 for details
Offset	13	0..8190 8191	1 meter	Offset of current position from the current path's start point modulo 8191. Value 8191 is invalid.
Position Index	2	0..3		If Av2HP supports positioning alternatives, contains index of this Position Candidate. Candidate 0 is always most probable position candidate, candidate 1 second etc. If Av2HP does not support positioning alternatives, it will be always 0. See also section 6.4.

Field	Length (bits)	Value Range	Units	Description
Position Age	9	0..510 511	5 millisecond	Time difference between the moment the message is sent and the moment the vehicle position is calculated by the positioning sub-system : • **0...509:** 0 … 2545 ms • **510:** > 2545 ms • **511:** N/A
Speed	9	0..510 511	0.2 m/s	Speed of the vehicle projected to the path. Values are: • **0:** ≤ -12.8 m/s • **1...63:** -12.6 …-0.2 m/s • **64:** Standing still • **65...509:** +0.2…+89.0 m/s • **510:** ≥ +89.2 m/s • **511:** N/A
Relative Heading	8	0..254 255	360/254 degrees	Heading relative to path: • **0:** In path direction • **1:** 1.417 degrees right • **2..126:** Right from path direction • **126:** 178.583 degrees right • **127:** Opposite direction • **128:** 178.583 degrees left (181.417 degrees right) • **129..252**: Left from path direction • **253**: 1.417 degrees left (358.583 degrees right) • **254**: Unknown • **255:** N/A

Field	Length (bits)	Value Range	Units	Description
Position Probability	5	0..30 31	100/30 percent.	Position Probability is calculated by Positioning module of ADASIS v2 Horizon Provider. Different Av2HPs will use different algorithms to calculate this value. Value 0 indicates *unknown*; Value 31 means *N/A*.
Position Confidence	3	0..6 7		Position confidence: this field contains Av2HP implementation specific values. Use 0 for highest confidence, 6 for lowest confidence. Value 7 means N/A
Current Lane	3	0..6 7		Current lane; for details, see Table 8: Current Lane values.
Reserved	1			

Table 5: POSITION message



	7	6	5	4	3	2	1	0
0	ADAS_Posn_MsgTyp 7 msb	6	5 lsb	ADAS_Posn_Offset 4 msb	3	2	1	0
1	ADAS_Posn_Offset 15	14	13	12	11	10	9	8 lsb
2	ADAS_Posn_CycCnt 23 msb	22 lsb	ADAS_Posn_PathIdx 21 msb	20	19	18	17	16 lsb
3	ADAS_Posn_Idx 31 msb	30 lsb	ADAS_Posn_PosProbb 29 msb	28	27	26	25 lsb	ADAS_Posn_Age 24 msb
4	ADAS_Posn_Age 39	38	37	36	35	34	33	32 lsb
5	47	ADAS_Posn_PosConfdc 46 msb	45	44 lsb	ADAS_Posn_CurLane 43 msb	42	41 lsb	ADAS_Posn_Spd 40 msb
6	ADAS_Posn_Spd 55	54	53	52	51	50	49	48 lsb
7	ADAS_Posn_RelHead 63 msb	62	61	60	59	58	57	56 lsb

Table 6: CAN layout of POSITION message

Value	Message	Description
0		Unknown
1	POSITION	Position not in Digitized Area
2	POSITION	Position not on-road
3	POSITION	Av2HP Vehicle Positioning sub-system not calibrated
4	SEGMENT	Messages with this path index refer only to the current vehicle position segment (for simple applications that do not need a prediction horizon - see description in section 5.9 "ADASIS Mini")
5	STUB	(= Sub-Path index field of STUB message) -- indicates the existence of a crossroad branch that will not further be appended by path data - see section 5.2.
6	STUB	(= Sub-Path index field of STUB message) -- Indicates that the STUB message is used to transmit the HEADING CHANGE and RELATIVE PROBABILITY of the main path at an intersection (see also section 5.2)
7		Reserved

Table 7: Path Indices with Special Meaning

Value	Description
0	Unknown
1	Emergency lane
2	Single-lane road
3	Left-most lane
4	Right-most lane
5	One of middle lanes on road with three or more lanes
6	Reserved
7	N/A

Table 8: Current Lane values

4.5 SEGMENT message

This message specifies attributes of the part of the road ahead. The SEGMENT message packs a number of the most important PROFILEs in one CAN frame.

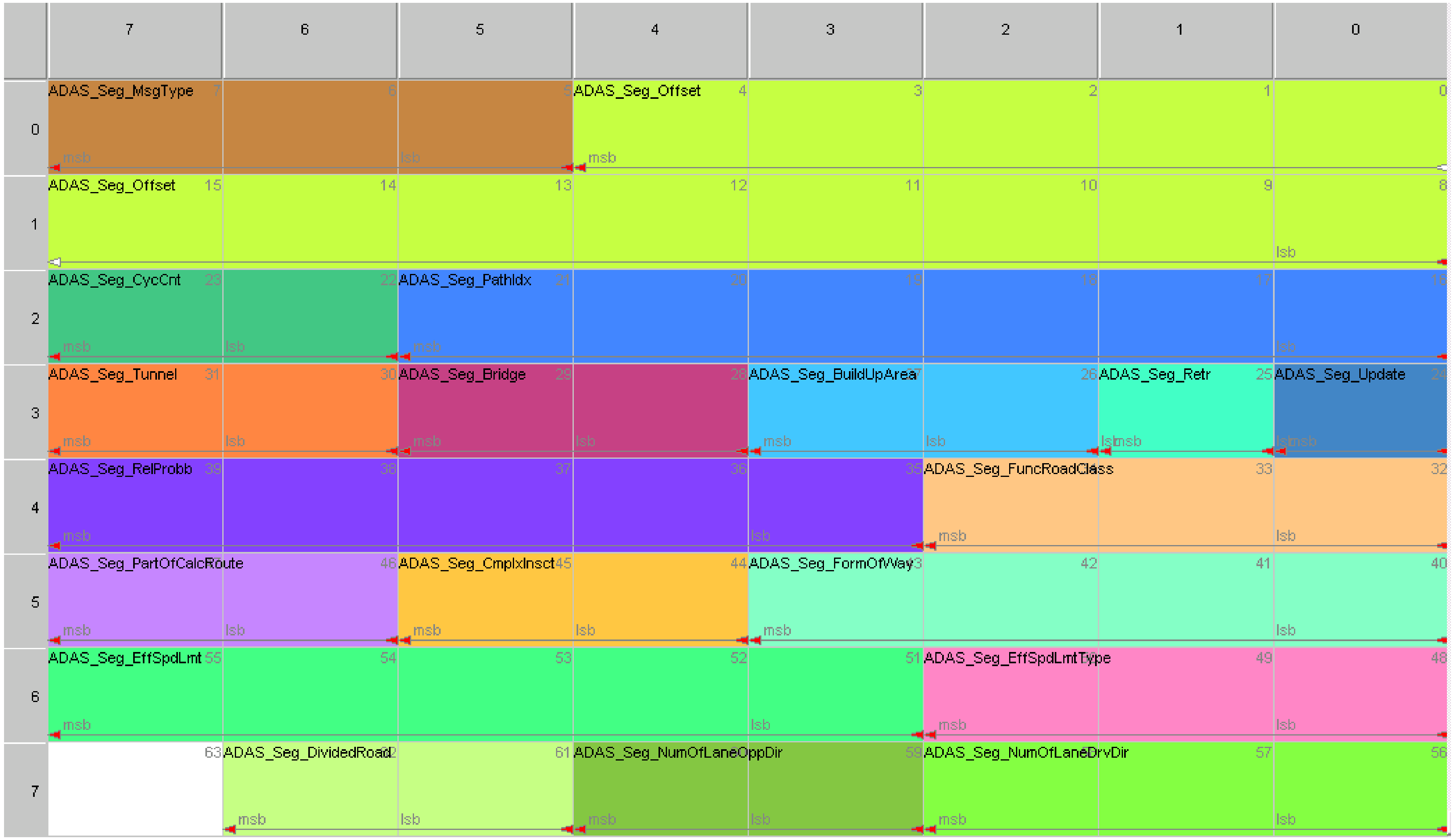
Field	Length (bits)	Value Range	Units	Description
Message Type	3	2		See Table 1
Cyclic Counter	2	0..3		Cyclic counter of this message type: Reconstructor can use this value to detect missing messages
Retransmission	1	0 (=false) 1 (=true)		If *true*, this message was sent already at least once before
Path Index	6	4 8..63		Value is Index of path of which this segment is part
Offset	13	0..8190 8191	1 meter	Offset of this segment from start position of the path modulo 8191; Value 8191 is invalid
Update	1	0 (=false) 1 (=true)		Value *true* indicates that this message contains updated information of already transmitted SEGMENT. See section 5.6 for details.
Functional Road Class	3	0, 1..6 7		Functional Road Class indicates relative importance of the road in the routing network and lower values indicate higher priority. This information is usually map-provider specific. Regular values are 1-6; value 0 is unknown, value 7 indicates N/A.

Field	Length (bits)	Value Range	Units	Description
Form of Way	4	0..14 15		Values are described in Table 11; see also [8] (page K-8)
Effective Speed Limit	5	0..30 31	km/h or mph	Current Speed Limit: May be implicit (for instance, in cities), or explicit. Current time of day, day of week and weather conditions can taken into account by Horizon Provider in determining effective speed limit. May be expressed as km/h or mph, see section 4.9 META-DATA Message. Note that this might be a rounded value. E.g. if there is a speed limit of 19 mph, it will be rounded to 20 mph, because 19 is not directly represented. Exact values should be transmitted via PROFILE messages. Nevertheless, in most cases this will be the exact value. Coding of this value described in Table 12.
Effective Speed Limit Type	3	0..6 7		Describes what kind of Speed Limit is in effect. See Table 13: Speed Limit Type.
Number of lanes in driving direction	3	0..6 7		Number of lanes in driving direction: also implies driving direction. 0 denotes one-direction road driven in wrong direction, 1-5 are verbatim values, and 6 indicate 6 or more lanes. Value 7 means *N/A*.
Number of lanes in opposite direction	2	0..2 3		Number of lanes in opposite direction: 0 denotes one-direction road, 1 for one lane only; 2 indicates 2 or more lanes. Value 3 means *N/A*.

Field	Length (bits)	Value Range	Units	Description
Tunnel	2	0..2 3		0: Segment is not part of a tunnel 1: Segment is part of a tunnel 2: Unknown 3: N/A.
Bridge	2	0..2 3		0: Segment is not part of a bridge. 1: Segment is part of a bridge. 2: Unknown 3: N/A
Divided Road	2	0..2 3		0: Segment is not part of a divided road/dual carriageway. 1: Segment is part of a divided road/dual carriageway. 2: Unknown 3: N/A
Built-up Area	2	0..2 3		0: Segment is not part of a built-up area. 1: Segment is part of a built-up area. 2: Unknown 3: N/A

Field	Length (bits)	Value Range	Units	Description
Complex Intersection	2	0..2 3		0: Segment is not part of complex intersection. 1: Segment is part of a complex intersection. 2: Unknown 3: N/A Complex intersection is defined by the GDF level 2 complex feature “Intersection” containing more than one junction and/or road element. "Complex intersection" shall be set when the vehicle drives on road elements with source node and target node belonging to the same complex feature (“plural junction” with "intersection internal" road elements)
Relative Probability	5	0..30 31	100/30 percent	Relative probability of this segment in relation to the previous segment on the same path; for the first segment on the path, this probability is equal as relative probability of the STUB, while for all subsequent segments till next crossing it will be 100%. Value 0 indicates that driving along this segment of the path is not allowed. Value 31 means unknown or *N/A*.
Part of Calculated Route	2	0..2		0: Segment is not part of Calculated Route 1: Segment is part of Calculated Route 2: Unknown 3: N/A
Reserved	1			

Table 9: SEGMENT message



	7	6	5	4	3	2	1	0
0	ADAS_Seg_MsgType 7 msb	6	5 lsb	ADAS_Seg_Offset 4 msb	3	2	1	0
1	ADAS_Seg_Offset 15	14	13	12	11	10	9	8 lsb
2	ADAS_Seg_CycCnt 23 msb	22 lsb	ADAS_Seg_PathIdx 21 msb	20	19	18	17	16 lsb
3	ADAS_Seg_Tunnel 31 msb	30 lsb	ADAS_Seg_Bridge 29 msb	28 lsb	ADAS_Seg_BuildUpArea 27 msb	26 lsb	ADAS_Seg_Retr 25 lsb msb	ADAS_Seg_Update 24 lsb msb
4	ADAS_Seg_RelProbb 39 msb	38	37	36	35 lsb	ADAS_Seg_FuncRoadClass 34 msb	33	32 lsb
5	ADAS_Seg_PartOfCalcRoute 47 msb	46 lsb	ADAS_Seg_CmplxInsct 45 msb	44 lsb	ADAS_Seg_FormOfWay 43 msb	42	41	40 lsb
6	ADAS_Seg_EffSpdLmt 55 msb	54	53	52	51 lsb	ADAS_Seg_EffSpdLmtType 50 msb	49	48 lsb
7	63	ADAS_Seg_DividedRoad 62 msb	61 lsb	ADAS_Seg_NumOfLaneOppDir 60 msb	59 lsb	ADAS_Seg_NumOfLaneDrvDir 58 msb	57	56 lsb

Table 10: CAN layout of SEGMENT message

Value	Description
0	Unknown
1	Freeway or Controlled Access road that is not a slip road/ramp
2	Multiple Carriageway or Multiply Digitized Road
3	Single Carriageway (default)
4	Roundabout Circle
5	Traffic Square/Special Traffic Figure
6	Reserved
7	Reserved
8	Parallel Road (as special type of a slip road/ramp)
9	Slip Road/Ramp on a Freeway or Controlled Access road
10	Slip Road/Ramp (not on a Freeway or Controlled Access road)
11	Service Road or Frontage Road
12	Entrance to or exit of a Car Park
13	Entrance to or exit to Service

Value	Description
14	Pedestrian Zone
15	N/A

Table 11: Form of Way (4-bit)

Value	Description
0	Unknown
1	≤ 5 km/h [mph]
2	7 km/h [mph]
3	10 km/h [mph]
4	15 km/h [mph]
5	20 km/h [mph]
6	25 km/h [mph]
7	30 km/h [mph]
8	35 km/h [mph]
9	40 km/h [mph]
10	45 km/h [mph]
11	50 km/h [mph]
12	55 km/h [mph]
13	60 km/h [mph]
14	65 km/h [mph]
15	70 km/h [mph]
16	75 km/h [mph]
17	80 km/h [mph]
18	85 km/h [mph]
19	90 km/h [mph]
20	95 km/h [mph]
21	100 km/h [mph]
22	105 km/h [mph]
23	110 km/h [mph]
24	115 km/h [mph]
25	120 km/h [mph]
26	130 km/h [mph]
27	140 km/h [mph]
28	150 km/h [mph]
29	160 km/h [mph]
30	Unlimited
31	N/A

Table 12: Effective Speed Limit (5-bit)

Value	Description
0	Implicit (for instance, default speed limit in the cities)
1	Explicit – on traffic sign
2	Explicit – by night
3	Explicit – by day
4	Explicit – time of day
5	Explicit – rain
6	Explicit – snow
7	Unknown

Table 13: Speed Limit Type (3-bit)

4.6 STUB message

The STUB message describes crossing on the path. Each stub is the origin for a new path.

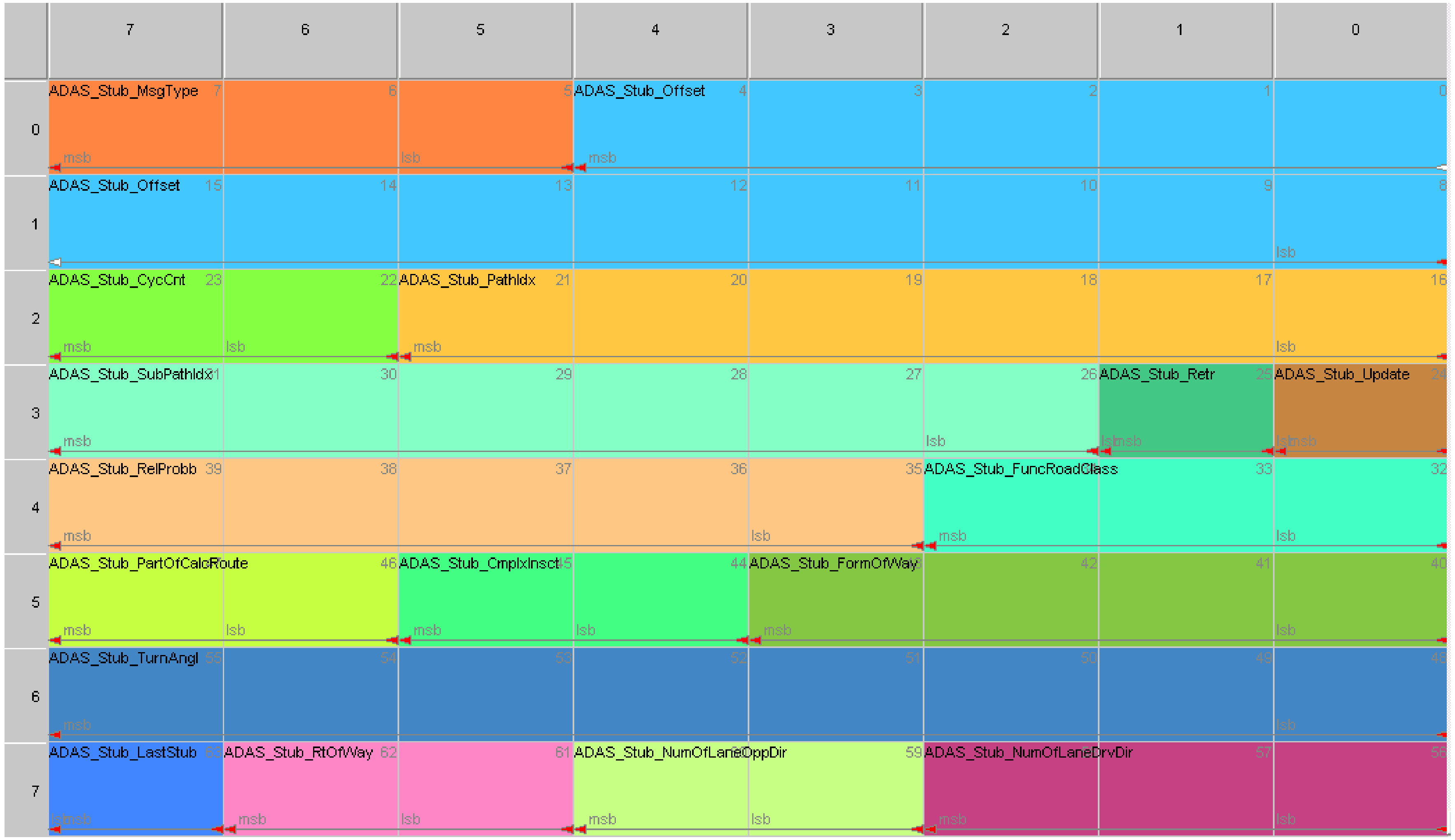
Field	Length (bits)	Value Range	Units	Description
Message Type	3	3		See Table 1.
Cyclic Counter	2	0..3		Cyclic counter of this message type: Reconstructor can use this value to detect missing messages.
Retransmission	1	0 (=false) 1 (=true)		If *true*, this message has been sent at least once before.
Path Index	6	0 8..63		Index of parent path; zero indicates this STUB starts first path in the Horizon.
Offset	13	0..8190 8191	1 meter	Offset of the position of the crossing from the start path position modulo 8191. Offset 8191 is invalid.
Update	1	0 (=false) 1 (=true)		Value *true* indicates that this message contains updated information of already transmitted STUB. See section 5.6 for details.
Sub-Path Index	6	5,6,8..63		This is Index of new path. See also section 5.2.

Field	Length (bits)	Value Range	Units	Description
Turn Angle	8	0..254 255	360/254 degrees	Angle of new path relative to parent path: • **0:** In path direction • **1:** 1.417 degrees right • **2..125:** Right from path direction • **126:** 178.583 degrees right • **127:** Opposite direction • **128:** 178.583 degrees left (181.417 degrees right) • **129..252**: Left from path direction • **253**: 1.417 degrees left (358.583 degrees right) • **254**: Unknown • **255:** N/A In general, this is angle between extension of incoming link and outgoing link. Exact method of calculation is not defined.
Relative Probability	5	0..30 31	100/30 percent	Relative probability of the new path on the crossing; sum of all relative stub probabilities (including continuation stub with sub-path index of 6) is 100. Value 0 indicates that turn is not allowed. Value 31 means unknown or *N/A*.
Form of Way	4	0..14 15		New path Form of Way; values are described in Table 11; see also [8] (page K-8)
Number of lanes in driving direction	3	0..6 7		Number of lanes in driving direction of new path: Also implies driving direction. 0 denotes one-direction road driven in wrong direction, 1-5 are verbatim values, and 6 indicate 6 or more lanes. Value 7 means *N/A*.

Field	Length (bits)	Value Range	Units	Description
Number of lanes in opposite direction	2	0..2 3		Number of lanes in opposite direction on new path: 0 denotes one-direction road, 1 for one lane only; 2 indicates two or more lanes. Value 3 means *N/A*.
Complex Intersection	2	0..2 3		0: New path is not part of complex intersection. 1: New path is part of a complex intersection. 2: Unknown 3: N/A Complex intersection is defined by the GDF level 2 complex feature “Intersection” containing more than one junction and/or road element. "Complex intersection" shall be set when the vehicle drives on road elements with source node and target node belonging to the same complex feature (“plural junction” with "intersection internal" road elements)
Right of Way	2	0..2 3		0: Parent path has right-of-way over sub-path 1: Sub-path has right-of-way over parent path 2: Unknown 3: N/A
Functional Road Class	3	0, 1..6 7		Functional Road Class of new path indicates relative importance of the road in the routing network and lower values indicate higher priority. This information is usually map-provider specific. Regular values are 1-6; value 0 is unknown, value 7 indicates N/A.

Field	Length (bits)	Value Range	Units	Description
Part of Calculated Route	2	0..2		0: New path from this point on is not part of Calculated Route 1: New path from this point on is part of Calculated Route 2: Unknown 3: N/A
Last Stub at Offset	1	0 (=false) 1 (=true)		Value *true* indicates that this STUB message is last one that refers to given parent path and offset; value *false* says that more STUB messages will be sent
Reserved	0			

Table 14: STUB message



	7	6	5	4	3	2	1	0
0	ADAS_Stub_MsgType 7 msb	6	5 lsb	ADAS_Stub_Offset 4 msb	3	2	1	0
1	ADAS_Stub_Offset 15	14	13	12	11	10	9	8 lsb
2	ADAS_Stub_CycCnt 23 msb	22 lsb	ADAS_Stub_PathIdx 21 msb	20	19	18	17	16 lsb
3	ADAS_Stub_SubPathIdx 31 msb	30	29	28	27	26 lsb	ADAS_Stub_Retr 25 lsb msb	ADAS_Stub_Update 24 lsb msb
4	ADAS_Stub_RelProbb 39 msb	38	37	36	35 lsb	ADAS_Stub_FuncRoadClass 34 msb	33	32 lsb
5	ADAS_Stub_PartOfCalcRoute 47 msb	46 lsb	ADAS_Stub_CmplxInsct 45 msb	44 lsb	ADAS_Stub_FormOfWay 43 msb	42	41	40 lsb
6	ADAS_Stub_TurnAngl 55 msb	54	53	52	51	50	49	48 lsb
7	ADAS_Stub_LastStub 63 lsb msb	ADAS_Stub_RtOfWay 62 msb	61 lsb	ADAS_Stub_NumOfLaneOppDir 60 msb	59 lsb	ADAS_Stub_NumOfLaneDrvDir 58 msb	57	56 lsb

Table 15: CAN layout of STUB message

4.7 PROFILE SHORT Message

This message contains two 10-bit *Profile Spots* that are up to 1023 meters away. Interpretation of spot values as well as profile interpolation between spot values depends on the actual *Profile Type*. For all parts of a path where a profile is not defined or available, a default value is assumed (also defined by the *Profile Type*).

Field	***Length (bits)***	***Value Range***	***Units***	***Description***
Message Type	3	4		See Table 1
Cyclic Counter	2	0..3		Cyclic counter of this message type and profile type: Reconstructor can use this value to detect missing messages
Retransmission	1	0 (=false) 1 (=true)		If *true*, this message was sent already at least once before
Path Index	6	8..63		Index of the path where this profile(s) is located
Offset	13	0..8190 8191	1 meter	Offset of position of the first profile spot from start position modulo 8191. Offset 8191 is invalid
Update	1	0 (=false) 1 (=true)		Value *true* indicates that this message contains updated information of already transmitted PROFILE SHORT. See section 5.6 for details.
Profile Type	5	1..31 0		See Table 17: Standard PROFILE SHORT Types and section 5.8
Control Point	1	0 (=false) 1 (=true)		If *true*, values in this message specifies additional control points required for the interpolation of this profile; if *false*, actual profile values are specified. See also section 5.8.

Field	Length (bits)	Value Range	Units	Description
Value 0	10	0..1022 1023		Profile value at *Offset;* value 1023 is unknown or N/A.
Distance 1	10	0..1022 1023	1 meter	Distance from first profile spot to the second one. If 0, no other profile values are defined in this message; offset 1023 is invalid.
Value 1	10	0..1022 1023		Profile value at *Offset+Distance1;* value 1023 is unknown or N/A.
Accuracy	2	0..3		Accuracy of profile value is implementation specific. 0 means highest accuracy. 2 mean lowest accuracy. 3 indicate that accuracy is unknown.
Reserved	0			There are no unused bits.

Table 16: PROFILE SHORT message

PROFILE Type	Description	Size (bits)	Multiplier	Offset	Value Interpretation	Interpolation Type	Default Value
0	N/A						
1	Curvature	10			See Table 30	Linear	1023 (unknown)

PROFILE Type	Description	Size (bits)	Multiplier	Offset	Value Interpretation	Interpolation Type	Default Value
2	Route Number Types	10			Bit-coded field; if bit is set, the corresponding Route number assigned to the road (see section 9.2): Bit 0: Street with no Route number Bit 1: Route number type1 Bit 2: Route number type 2 Bit 3: Route number type 3 Bit 4: Route number type 4 Bit 5: Route number type 5 Bit 6: Route number type 6 Bit 7: Reserved (set to 0) Bit 8: Reserved (set to 0) Bit 9: Reserved (set to 0) See also [8] and Table 31	Step	0 (N/A)
3	Slope	10	0.1	-51.1	Slope in percent.	Step	1023 (unknown)
4	Slope	10	0.1	-51.1	Slope in percent.	Linear	1023 (unknown)

PROFILE Type	Description	Size (bits)	Multiplier	Offset	Value Interpretation	Interpolation Type	Default Value
5	Road accessibility	10			Bit-coded field; if bit is set, road is accessible for particular class of actors: Bit 0: Passenger Cars Bit 1: Pedestrians Bit 2: Bus Bit 3: Delivery Bit 4: Emergency Bit 5: Taxi Bit 6: Trough Traffic Bit 7: Trucks Bit 8: Reserved (set to 0) Bit 9: Reserved (set to 0)	Step	0 (no access)
6	Road condition	10			Low 4 bits (0-3) are coded as follows: 1xxx: condition good 0xxx: condition poor x000: rigid (paved) x100: flexible (paved) x010: blocks (unpaved) x110: gravel (unpaved) x001: dirt (unpaved) 1111: unknown Upper 6 bits (9-4) are reserved for future use (set to 111111).	Step	1023 (unknown)

PROFILE Type	Description	Size (bits)	Multiplier	Offset	Value Interpretation	Interpolation Type	Default Value
7	Variable Speed Sign Position	10			Bit Coded Field, see Table 18	Spot (Attachment)	0 (unknown)
8	Heading Change	10	360/254 degrees		Heading change relative to path: • **0:** In path direction • **1:** 1.417 degrees right • **2..126:** Right from path direction • **126:** 178.583 degrees right • **127:** Opposite direction • **128:** 178.583 degrees left (181.417 degrees right) • **129..252:** Left from path direction • **253**: 1.417 degrees left (358.583 degrees right) • **254**: Unknown • **≥ 255:** N/A Exact method of calculation of this value is implementation specific. Units are chosen to be the same as *Relative Heading* field of POSITION message and *Turn Angle* of STUB message.	Spot (Attachment)	254 (unknown)
9-15	Reserved for standard types						

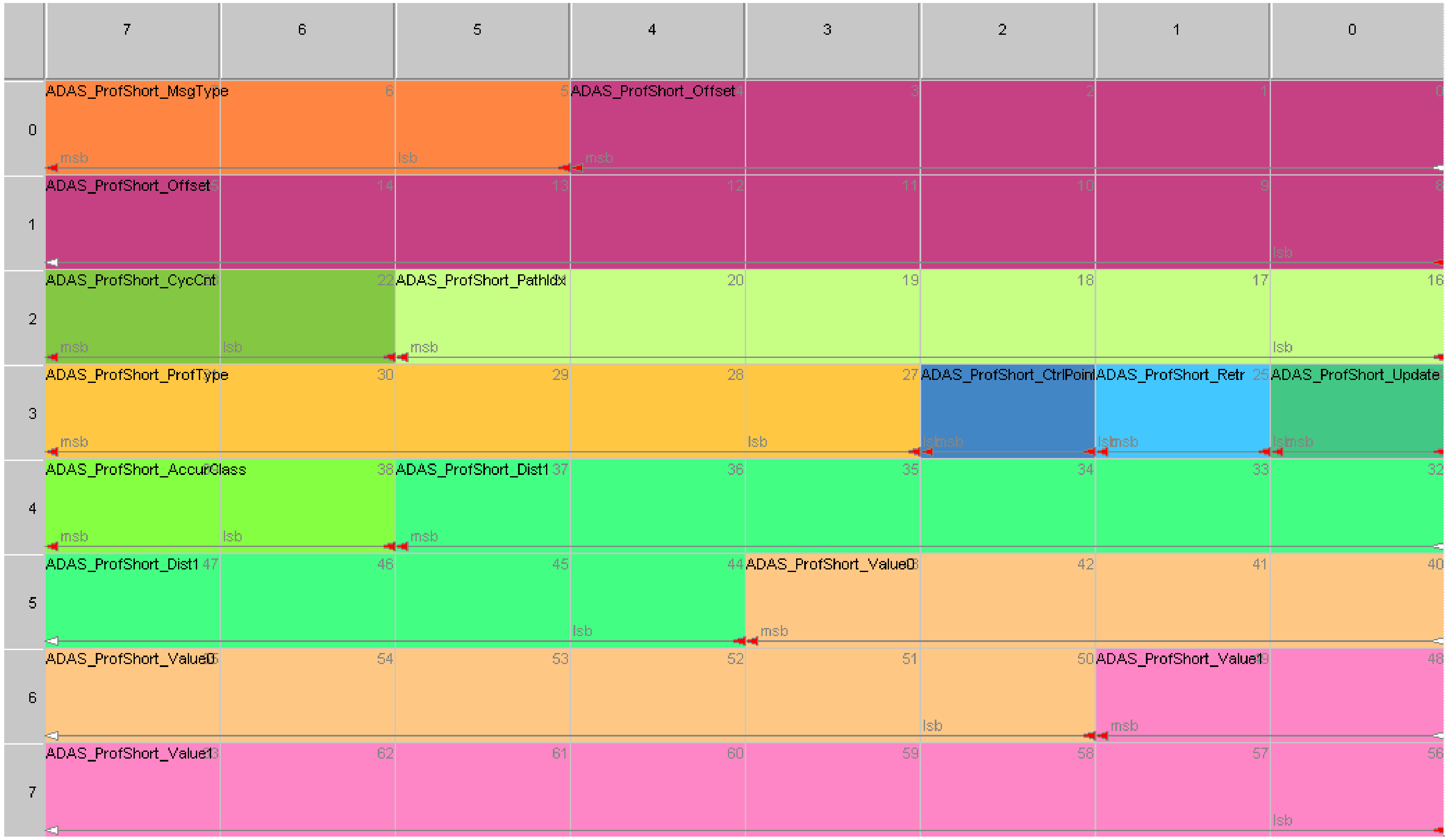
PROFILE Type	Description	Size (bits)	Multiplier	Offset	Value Interpretation	Interpolation Type	Default Value
16-31	Reserved for system specific types						

Table 17: Standard PROFILE SHORT Types

Bit	Description
0	Position of Variable Speed Sign is not known/defined
1	Variable Speed Sign is located left of the road
2	Variable Speed Sign is located right of the road
3	Variable Speed Sign is located above road, but lane is unknown/undefined
4	Variable Speed Sign is located above Emergency Lane
5	Variable Speed Sign is located above 1st lane from the right (left, in left-driving countries)
6	Variable Speed Sign is located above 2nd lane from the right (left, in left-driving countries)
7	Variable Speed Sign is located above 3rd lane from the right (left, in left-driving countries)
8	Variable Speed Sign is located above 4th lane from the right (left, in left-driving countries)

Bit	Description
9	Variable Speed Sign is located above 5th lane from the right (left, in left-driving countries)

Table 18: Variable Speed Sign Positions



	7	6	5	4	3	2	1	0
0	ADAS_ProfShort_MsgType 7 msb	6	5 lsb	ADAS_ProfShort_Offset 4 msb	3	2	1	0
1	ADAS_ProfShort_Offset 15	14	13	12	11	10	9	8 lsb
2	ADAS_ProfShort_CycCnt 23 msb	22 lsb	ADAS_ProfShort_PathIdx 21 msb	20	19	18	17	16 lsb
3	ADAS_ProfShort_ProfType 31 msb	30	29	28	27 lsb	ADAS_ProfShort_CtrlPoint 26 lsb msb	ADAS_ProfShort_Retr 25 lsb msb	ADAS_ProfShort_Update 24 lsb msb
4	ADAS_ProfShort_AccurClass 39 msb	38 lsb	ADAS_ProfShort_Dist1 37 msb	36	35	34	33	32
5	ADAS_ProfShort_Dist1 47	46	45	44 lsb	ADAS_ProfShort_Value0 43 msb	42	41	40
6	ADAS_ProfShort_Value0 55	54	53	52	51	50 lsb	ADAS_ProfShort_Value1 49 msb	48
7	ADAS_ProfShort_Value1 63	62	61	60	59	58	57	56 lsb

Table 19: CAN layout of PROFILE SHORT

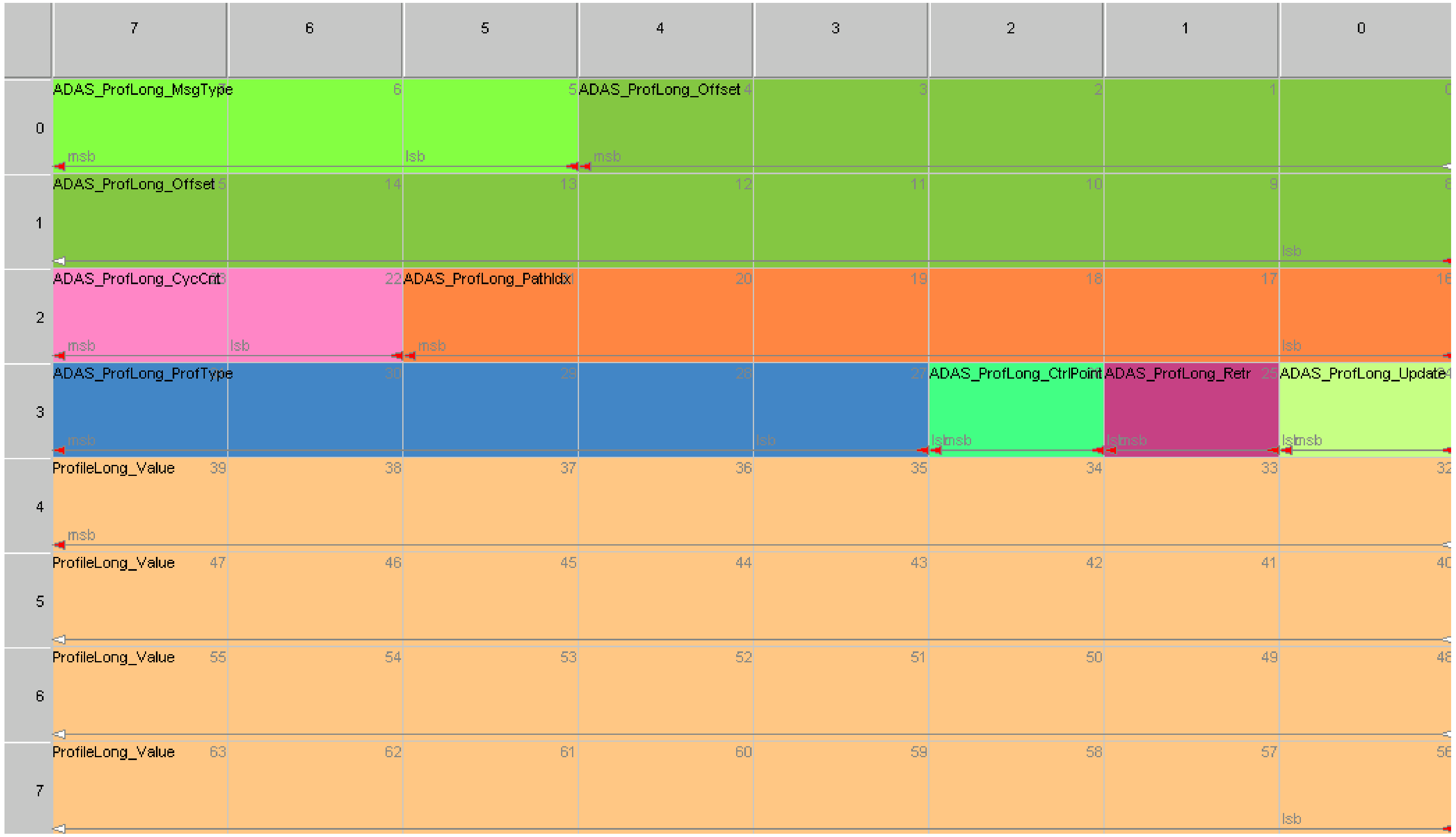
4.8 PROFILE LONG message

This message contains one 32-bit *Profile Spot*. Interpretation of spot values as well as profile interpolation between spot values depends on the actual *Profile Type*. For all parts of a path where the profile is not defined or available, a default value is assumed (also defined by the *Profile Type*).

Field	*Length (bits)*	*Value Range*	*Units*	*Description*
Message Type	3	5		See Table 1.
Cyclic Counter	2	0..3		Cyclic counter of this message type and profile type: Reconstructor can use this value to detect missing messages.
Retransmission	1	0 (=false) 1 (=true)		If *true*, this message was sent already at least once before.
Path Index	6	8..63		Index of the path where this profile(s) is located
Offset	13	0..8190 8191	1 meter	Offset of position of the first profile spot from start position modulo 8191. Offset 8191 is invalid.
Update	1	0 (=false) 1 (=true)		Value *true* indicates that this message contains updated information of already transmitted PROFILE LONG. See section 5.6 for details.
Profile Type	5	1..31 0		See Table 22: Standard PROFILE LONG Types and section 5.8.
Control Point	1	0 (=false) 1 (=true)		If *true*, values in this message specifies additional control points required for the interpolation of this profile; If *false*, actual profile values are specified. See also section 5.8.

Field	Length (bits)	Value Range	Units	Description
Value	32	0..2^{32}-2 2^{32}-1		Profile value at *Offset*, value 2^{32} -1 is unknown or N/A.
Reserved	0			There are no unused bits.

Table 20: PROFILE LONG message



	7	6	5	4	3	2	1	0
0	ADAS_ProfLong_MsgType 7 msb	6	5 lsb	ADAS_ProfLong_Offset 4 msb	3	2	1	0
1	ADAS_ProfLong_Offset 15	14	13	12	11	10	9	8 lsb
2	ADAS_ProfLong_CycCnt 23 msb	22 lsb	ADAS_ProfLong_PathIdx 21 msb	20	19	18	17	16 lsb
3	ADAS_ProfLong_ProfType 31 msb	30	29	28	27 lsb	ADAS_ProfLong_CtrlPoint 26 lsb msb	ADAS_ProfLong_Retr 25 lsb msb	ADAS_ProfLong_Update 24 lsb msb
4	ProfileLong_Value 39 msb	38	37	36	35	34	33	32
5	ProfileLong_Value 47	46	45	44	43	42	41	40
6	ProfileLong_Value 55	54	53	52	51	50	49	48
7	ProfileLong_Value 63	62	61	60	59	58	57	56 lsb

Table 21: CAN layout of PROFILE LONG message

PROFILE Type	Description	Size	Multiplier	Offset	Value Interpretation	Interpolation Type	Default Value
0	N/A						
1	Longitude	32 bit	0.0000001	-180	Longitude in degrees	Linear	$2^{32}-1$ (N/A)
2	Latitude	32 bit	0.0000001	-90	Latitude in degrees	Linear	$2^{32}-1$ (N/A)
3	Altitude	32 bit	0.01	-1000	Altitude in meters Allowed range: -1000m to +10000m	Linear	$2^{32}-1$ (N/A)
4	(Bézier) Control Point Longitude	32 bit	0.0000001	-180	Longitude in degrees; see also section 8.4.	Linear	$2^{32}-1$ (N/A)
5	(Bézier) Control Point Latitude	32 bit	0.0000001	-90	Latitude in degrees; see also section 8.4.	Linear	$2^{32}-1$ (N/A)
6	(Bézier) Control Point Altitude	32 bit	0.01	-1000	Altitude in meters; see also section 8.4. Allowed range: -1000m to +10000m	Linear	$2^{32}-1$ (N/A)
7	Link Identifier	32 bit			Link Identifier; see also section 8.5.	Step	$2^{32}-1$ (N/A)

PROFILE Type	Description	Size	Multiplier	Offset	Value Interpretation	Interpolation Type	Default Value
8-15	Reserved for standard types						
16-31	Reserved for system specific types						

Table 22: Standard PROFILE LONG Types

4.9 META-DATA Message

This message is sent with low frequency and it describes basic characteristics of the system or semi-permanent attributes of current position.

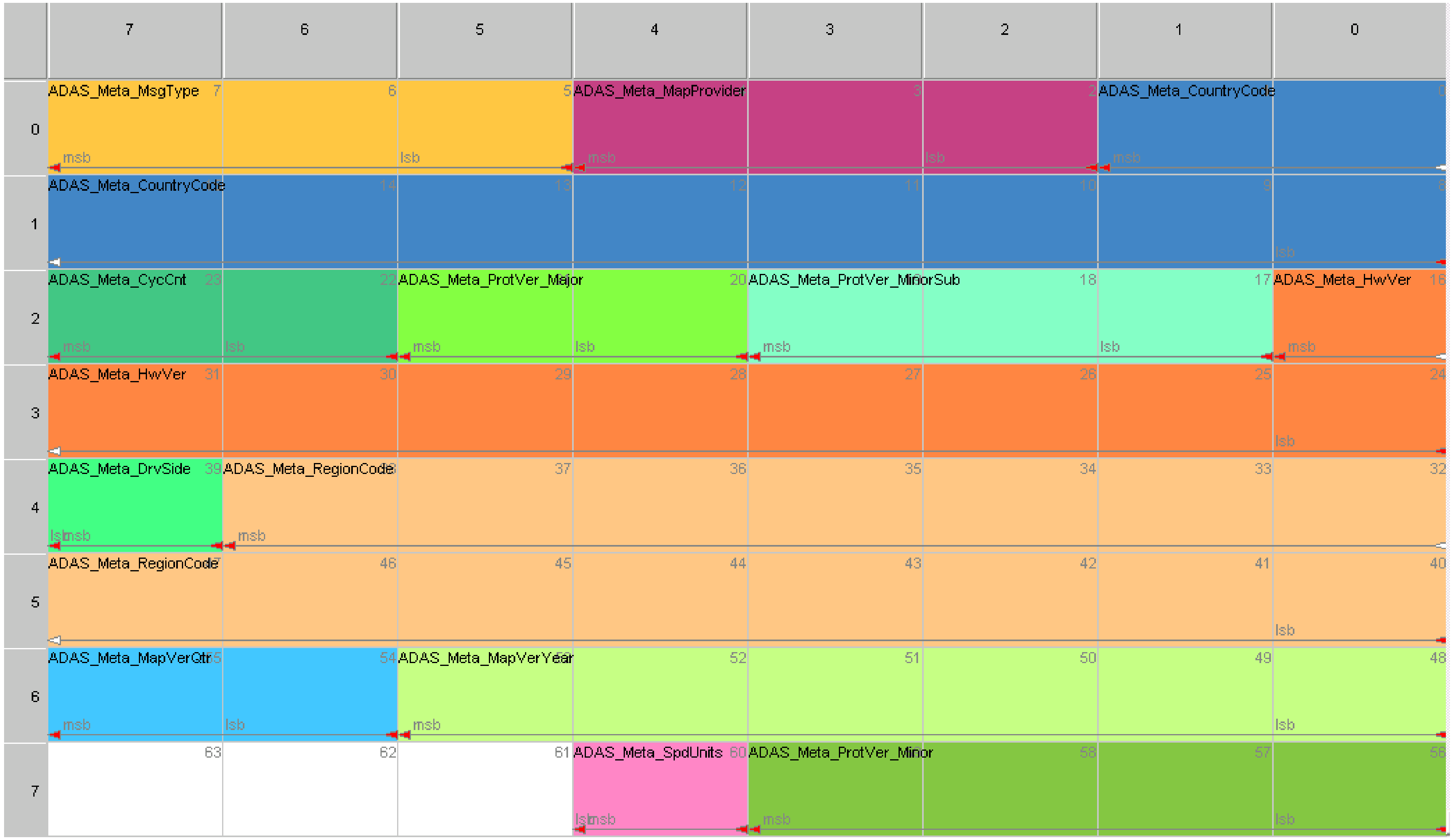
Field	Length (bits)	Value Range	Units	Description
Message Type	3	6		See Table 1.
Cyclic Counter	2	0..3		Cyclic counter of this message type: Reconstructor can use this value to detect missing messages.
Country Code	10	0..1023		ISO 3166-1 Release 2007-03 country code of current position; 0 stands for *unknown*. Valid values are those specified by ISO 3166-1 numeric [15].

Field	Length (bits)	Value Range	Units	Description
Region Code	15	0..32766 32767		For a given *Country Code*, specifies region of a country where vehicle is currently positioned. 0 stands for *unknown,* value 32767 means N/A All other values are based on ISO 3166-2 [16] with the code mapping scheme in chapter 9.4.
Driving Side	1	0, 1		0 stands for *Driving Side Left.* 1 stands for *Driving Side Right.* Default is 1.
Speed Units	1	0, 1		1 stands for *miles per hour (mph).* 0 stands for *kilometers/hour (km/h)*. Default is 0.
Major Protocol Version	2	0..3		Contains Major protocol version; See section 2.1 for details.
Minor Protocol Version	4	0..15		Contains Minor protocol version; See section 2.1 for details.
Minor Protocol Sub-Version	3	0..7		Contains Minor protocol sub-version; See section 2.1 for details.
Hardware Version	9	0..511		This field is ADAS Horizon Provider specific. 0 stands for *unknown*.

Field	Length (bits)	Value Range	Units	Description
Map Provider	3	0..6 7		Map Provider of current area (map tile, region, country …). • **0:** Unknown, • **1:** NAVTEQ, • **2:** TeleAtlas, • **3:** Zenrin, • **4:** reserved, • **5:** reserved, • **6:** reserved, • **7:** N/A. Please note if the whole map may be partitioned into parts that can be released on different times or even by different providers. In that case, value if this field may vary between messages.
Map Version (Year)	6	0..62 63	Year-2000 mode 63	Content of this field is Year of release of map of current area. For 2007, this field will contain value 7. Please note if the whole map may be partitioned into parts that can be released on different times or even by different providers. In that case, value of this field may vary between messages. 63 stands for *N/A*

Field	Length (bits)	Value Range	Units	Description
Map Version (Year Quarter)	2	0..3	Quarter-1	Content of this field is Year quarter of release of map of current area. For example, NAVTEQ Q4/2007 map will be described as *MAP PROVIDER = 1* *MAP VERSION (Y) = 7* *MAP VERSION (Q) = 3* Please note if the whole map may be partitioned into parts that can be released on different times or even by different providers. In that case, value if this field may vary between messages.
Reserved	3			

Table 23: META-DATA message



	7	6	5	4	3	2	1	0
0	ADAS_Meta_MsgType 7 msb	6	5 lsb	ADAS_Meta_MapProvider msb	3	2 lsb	ADAS_Meta_CountryCode msb	0
1	ADAS_Meta_CountryCode	14	13	12	11	10	9	8 lsb
2	ADAS_Meta_CycCnt 23 msb	22 lsb	ADAS_Meta_ProtVer_Major msb	20 lsb	ADAS_Meta_ProtVer_MinorSub msb	18	17 lsb	ADAS_Meta_HwVer 16 msb
3	ADAS_Meta_HwVer 31	30	29	28	27	26	25	24 lsb
4	ADAS_Meta_DrvSide 39 lsb msb	ADAS_Meta_RegionCode msb	37	36	35	34	33	32
5	ADAS_Meta_RegionCode	46	45	44	43	42	41	40 lsb
6	ADAS_Meta_MapVerQtr msb	54 lsb	ADAS_Meta_MapVerYear msb	52	51	50	49	48 lsb
7	63	62	61	ADAS_Meta_SpdUnits 60 lsb msb	ADAS_Meta_ProtVer_Minor msb	58	57	56 lsb

Table 24: CAN layout of META-DATA message

5 *Use of ADASIS v2 Protocol*

5.1 Required messages

Which messages the Av2HP will actually send in a given configuration is up to the application set up.

- POSITION message is the only one that is actually necessary to be sent in any configuration, since it defines vehicle position on the ADASIS v2 Horizon. In ADASIS v2 Mini (section 5.9), only required message is SEGMENT.
- STUB message has several roles in the protocol:
  1. It defines the parent/child relationship between paths;
  2. It marks the start of a new path;
  3. It describes attributes of the first part of the path (number of lanes, for instance)
  4. A collection of several STUBs at the same path and same offset describes an intersection.

     If none of the information above is required by any client application in the network, there is no need for the Av2HP to generate and send STUB messages. However, in that case the start of a new path must be implicitly recognized by the Client. ADASIS v2 defines such mechanism only for the single-path horizon case (section 3.3).
- SEGMENT message contains most important profiles packed together. If neither of those profiles is used by any application, the Av2HP does not need to send SEGMENTs.
- PROFILE messages will be sent if specific applications have need for the corresponding profile data.
- META-DATA message is not required to send if proper set of defaults are hard-coded on the client side. META-DATA, however, contains a number of semi-static fields (country, region, speed limit units …) that may change during driving as well as basic version information of the Horizon Provider. If any application uses this information, the META-DATA message must be transmitted by the Av2HP.

5.2 STUB types

STUB messages describe the start of a new path that has its origin on an existing one.

Fields PATH INDEX and OFFSET define the exact position on the existing path where new one starts. As such, crossings are defined as several STUBs at the same OFFSET of the PATH INDEX. In any case, PATH INDEX and SUB-PATH INDEX of STUB message must be different. SUB-PATH INDEX may be (see also Table 7: Path Indices with Special Meaning):

- **Value 5**: For these STUBs, no physical path is created, i.e. no messages will be appended. This Stub type is typically used for systems with very low bandwidth or storage capacity or for very low probability branches at intersections, barely to indicate their existence. If the vehicle turns to a stub with Path Index 5, the Av2HP must initiate a new path. Typically, a STUB message with a SUB-PATH INDEX value of 5 will be sent by Av2HPs providing only the 'single-path-horizon-with-stubs'.
- **Value 6**: As opposed to normal STUBs connecting side-branches to the main path, a STUB with a SUB-PATH INDEX of 6 provides information for the continuation of the main path such as TURN ANGLE and RELATIVE PROBABILTY. Most of the content of such a STUB message (such as NUMBER OF LANES) may be redundant as there is already a SEGMENT message with the same data. We will refer to this kind of stub as a *continuation stub*.
- **Values 8 to 63**: “Normal” STUB message, defining the origin of a new path; subsequent messages referring to this Index value will provide more data about it.

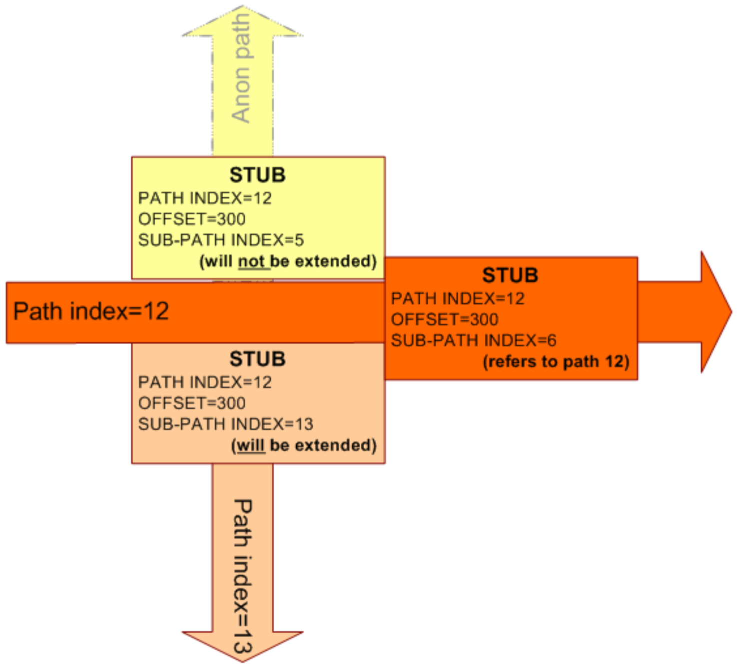


Figure 18: Three types of STUB message

5.3 Implicit STUBs

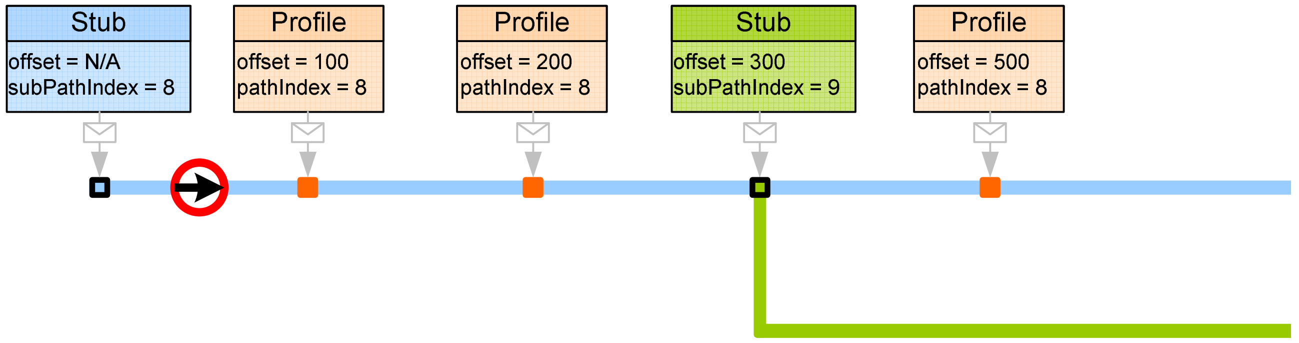
In general, the Av2HP uses STUB messages to inform the Av2HR that a new path entity is to be created. In that case, POSITION, SEGMENT and PROFILE messages will refer to a path index that was already contained as sub-path index in a previously sent STUB message.
The POSITION message may refer to path index that was not previously created using a STUB; in other words, at the very beginning, the Av2HP may send a POSITION as the first message in the communication. It should be avoided, however, to have a POSITION message for a path that only will be introduced in a later STUB message. Because the POSITION path index may be reused before receiving the corresponding STUB message, the application may incorrectly position the vehicle on the 'previous' path.

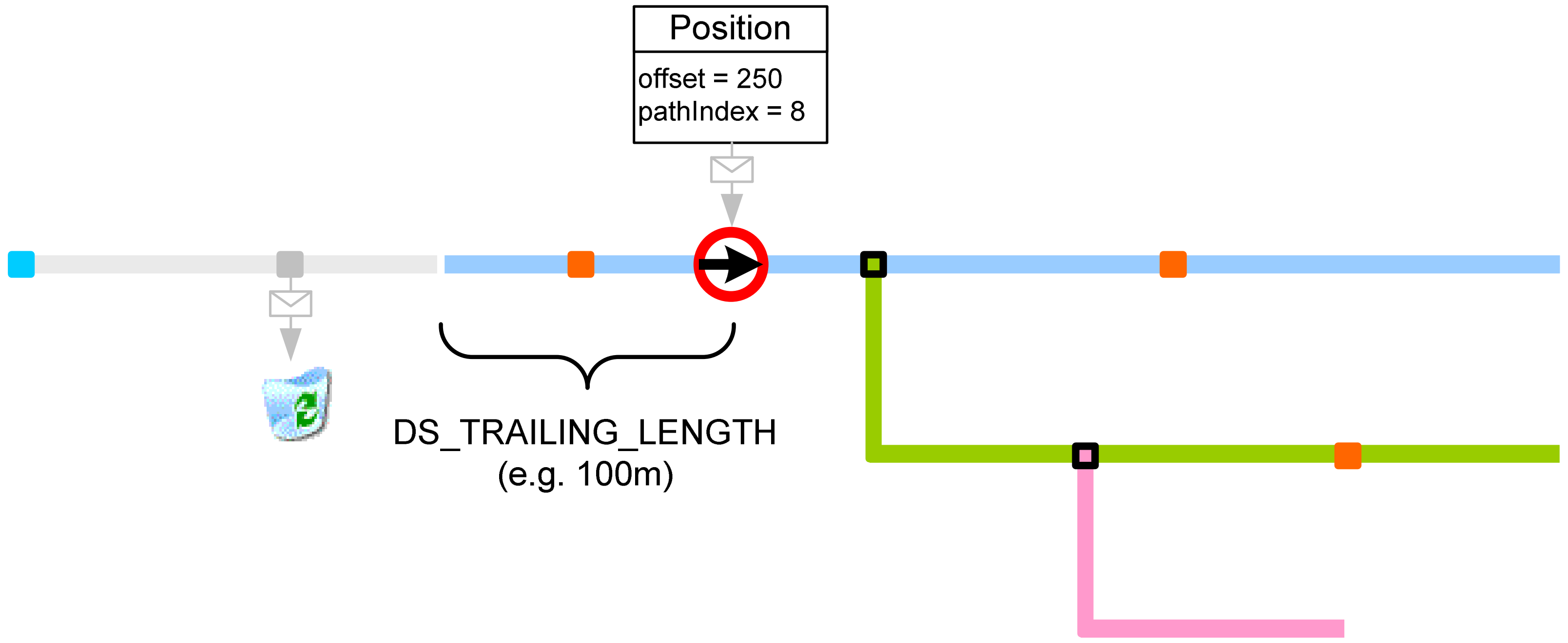
If the Av2HP is configured to only send a single-path horizon, the path entity may be created implicitly at the Av2HR when POSITION, SEGMENT or PROFILE messages refer to a new path index. As a result, the single-path horizon will use only two path indices, say 8 and 9. While the vehicle is driving the original (most-probable) path, all messages will use one same path index (say 8). From the moment when POSITION, SEGMENT or PROFILE messages refer to a different path index (say 9), the Av2HR shall conclude that the vehicle has left the original path, abandon all information of the 'old' path and start to transmit data for the new one.

5.4 Implicit entity removal on client

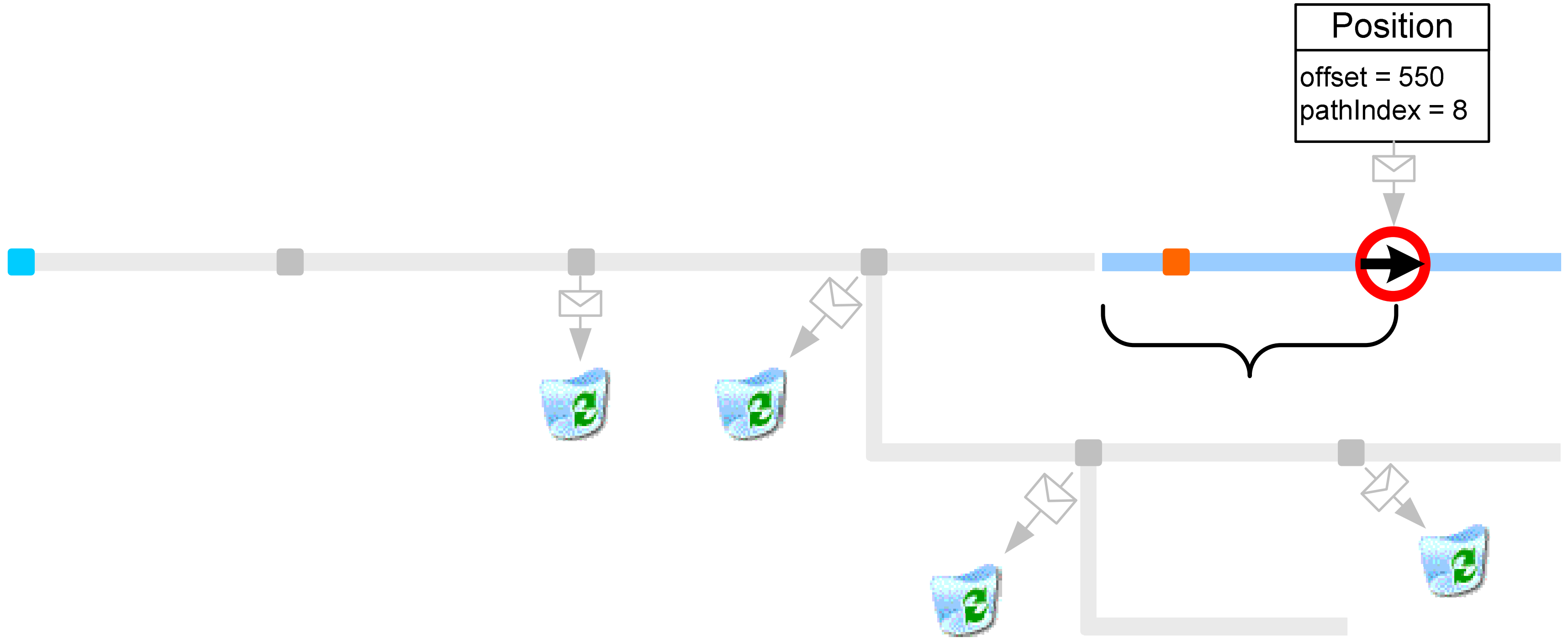
This section describes how the AV2HR handles deletion of profiles and stubs behind the current vehicle position. The guidelines also show how the AV2HP must handle special use cases to be consistent with the reconstructor content to avoid "memory leaks".

As a precondition, the AV2HP builds up a path with several profiles and one sub path. It is recommended to start creating the horizon at a positive (non-zero) offset along the first path, as some applications may always look behind the current position. This is particularly important at start-up, as the current vehicle position is still inaccurate and could therefore jump backwards.

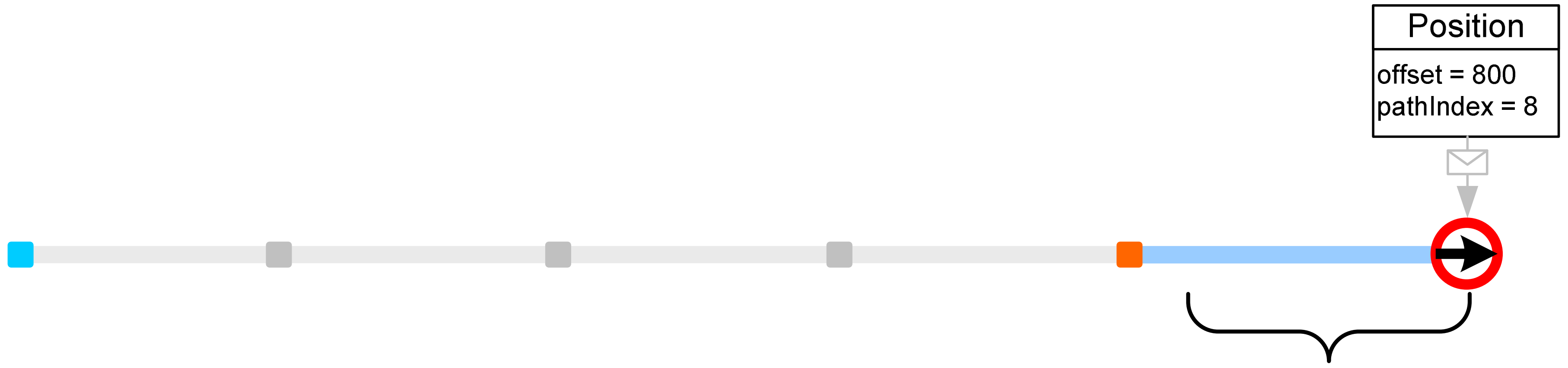




As the vehicle moves along the path, the AV2HP sends further profiles and one sub-path. The AV2HR should have a defined trailing length value (called, for instance, DS_TRAILING_LENGTH), which defines the distance back from the vehicle before entities can be deleted from the data store of AV2HR. This value must be consistent with the corresponding setting of the AV2HP.

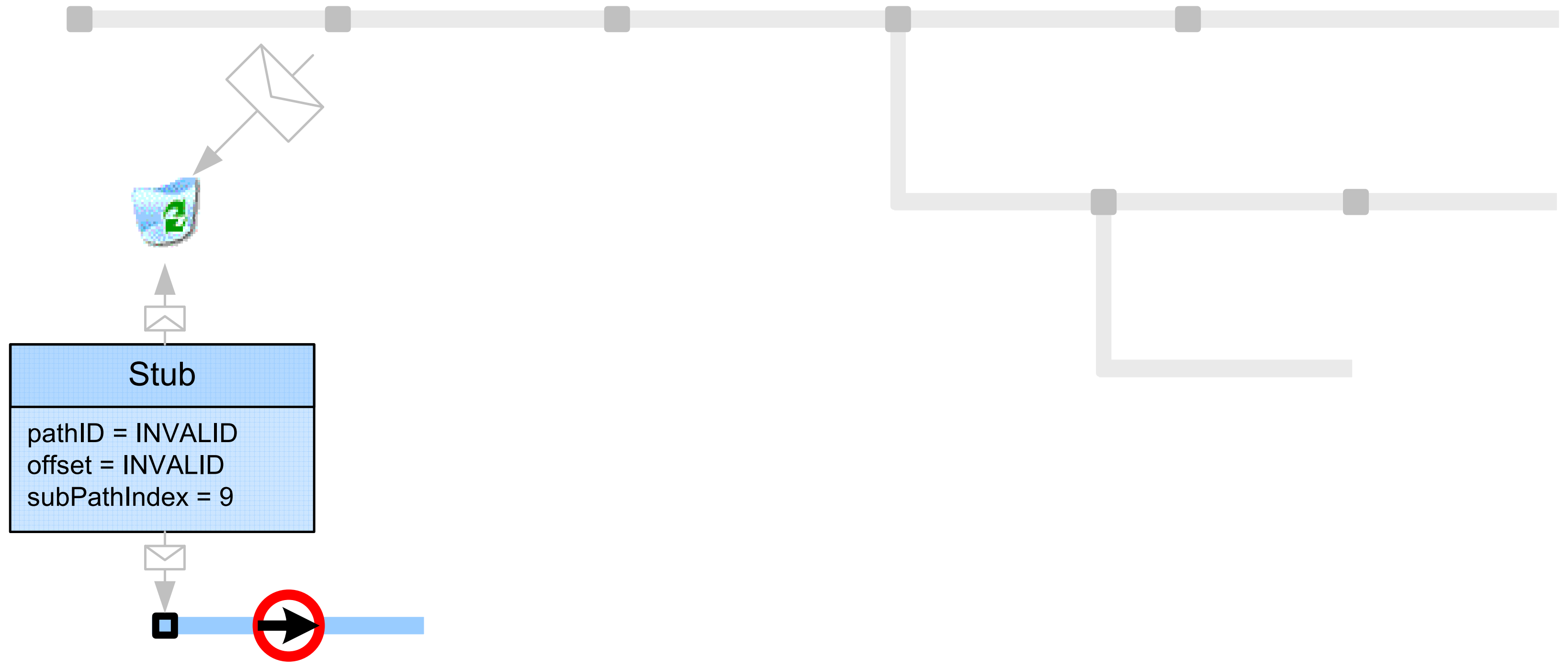


Another use case is when the vehicle passes a junction as defined by stubs. Not only the profiles and segments behind the vehicle can be removed, but since all sub-paths reachable from these stubs are no longer reachable by the vehicle, they can also be removed.



The stub information of the current path and the last valid profile behind the vehicle are two exceptions. Stub information will not be removed as long as the vehicle stays on this path, because it is needed for the application to retrieve path information concerning the current path. The last valid profile will also not be removed, because this could define the current profile value at the vehicle offset (e.g. current speed limit).

At this point it is important to ensure that, apart from transmission latencies, the provider's and the reconstructors view of the current horizon shall not be different. The system setup must ensure that the reconstructor does not run out of memory while the provider is still sending data.



To completely delete the reconstructors content, the provider shall send a new stub with pathID == INVALID and offset == INVALID. With this information, the reconstructor will completely initialize its data. The provider should always send this special initial stub before the horizon is built up from scratch. This could be the very first time after system start-up, if the vehicle position cannot be matched to the ADAS Horizon, or on request, e.g. by a complete retransmission request of the AV2HR application. This is especially useful if the ADAS Horizon application is starting up later than the AV2HP, or if the application gets an out-of-memory error code from the reconstructor. A retransmission request mechanism is not specified by the ADASIS protocol and must be defined system-specifically.

5.5 ADAS Horizon Path Creation in case of Vehicle Position Jumps and U-Turns

5.5.1 Backward Jump of Position Offset

The current vehicle position may jump backwards because of positioning and map-matching errors.

To allow backward position jumps it is recommended to start creating a new ADAS horizon in the Av2HP at a certain distance behind the current vehicle position. Moreover, when removing ADAS horizon entities behind the vehicle, it is recommended to keep all entities in the Av2HP and all ADASIS v2 horizon reconstructor unto a certain distance behind the current vehicle position.

If the vehicle position jumps backwards more than this certain distance, the Av2HP deletes the current path completely and must start a new path transmission also containing the new vehicle position. This new path is *not* connected to the former path by a STUB message. The allowed distance to jump backwards shall be set to an appropriate value in order to avoid frequent path deletion and creation steps.

5.5.2 Vehicle makes a U-Turn Manoeuvre

If a vehicle changes the direction in which a road is driven, the position offset of the vehicle with respect to the current path will decrease. When detecting a driving direction change the Av2HP shall delete the current path and create a new path in the new driving direction. This new path is *not* connected to the former path by a STUB message.

5.5.3 Vehicle drives backward

Normally the position offset of a vehicle on the ADAS horizon will increase in time. If a vehicle drives backward, the position offset will decrease until the maximum backward jump distance has been reached.

After that this situation can be handled as a U-turn manoeuvre creating a new path in the moving direction of the vehicle.

5.6 Updating of content after transmission

Most of the information that is transferred in SEGMENT, STUB and PROFILE messages is read from the digital map database and is of static nature in the sense that it does not change once it is transmitted. However, some of the data can change dynamically during the lifetime of the path. For instance:

- *Part of Calculated Route* SEGMENT attribute may change its value if the driver diverts from the original route and the system recalculates another one which follows a different path.
- *Relative Probability* of the STUB may change if the ADAS Horizon is constructed incrementally and new path data is revealed that influences probability calculations; for instance, when new route is calculated.
- SEGMENT *Effective Speed Limit* and *Effective Speed Limit Type* may change because of changed weather conditions; sudden rain, for instance;
- The PROFILE message may also contain Speed Limit or other sign information that depends on external factors such as weather conditions.

When changes are detected, Av2HP will inform the clients about new values of previously transmitted data.

To accomplish an update of the data on the client, *Update* flag of SEGMENT, STUB and PROFILE messages must be set to *true*; so that the client can correctly identify newly received messages as updates of already received ones, and not as new data that is to be inserted into the existing path network.

Depending on the configuration of how the Av2HP deals with the limited distance coding (see section 3.4.4), the OFFSET of updated data is to be interpreted differently:

- In case of unlimited paths / cyclic offsets, the OFFSET specified in the message must be interpreted as the relative offset from the current vehicle position (i.e. last POSITION message) rather than the absolute offset along the path modulo 8191.
  Such interpretation of OFFSET limits updates only to a distance of around 8 kilometers ahead of the vehicle, and only to the objects along the current path. However, since the vehicle is moving, the “window” of update is also moving forward and objects that are currently not updateable will become so as vehicle moves towards them.
- In case of limited paths / non-cyclic offsets, OFFSET is the absolute value along the path.

5.7 Solutions for Unlimited Horizons

As we have seen earlier, when implementing the Av2HP and Av2HR one must take care of the 6-bit limit of the PATH INDEX and the 13-bit limit for OFFSETs that are part of ADASIS v2 messages, if unlimited paths lengths and larger number of paths are required. This section describes how to deal with this issue.

5.7.1 Absolute path identifiers and path indices

If the ADAS sender or client needs an unlimited number of paths, this chapter describes a solution how to deal with this.

In the ADASIS v2 Protocol messages, the PATH INDEX and SUB-PATH INDEX fields contain the identification of the path. The 6-bit field size (minus 8 reserved values) limits the number of path indices to 56. In most cases, the ADASIS v2 Horizon will consist of less than 56 paths; however, during the vehicle travel, some paths will become obsolete (those behind the vehicle), while new paths will appear on the horizon front. Consequently, during normal driving a total of much more than 56 paths will come to life and vanish again

Internally on the provider and reconstructor side, each of those paths may have been represented by a code, having 16 or even 32 bit length, resulting in a virtually unique identifier which we will call the *absolute path identifier.*

As opposed to an internally used, ‘unique’ identification scheme, the ADASIS v2 messages only allow to distinguish 56 different paths as mentioned above. This limited path-identification used in the CAN Message domain is called *path index*. While the absolute path identifier is virtually unlimited and unique (i.e. only used once), a path index will be frequently reused and refer to different paths during a driving session.

The Av2HR and Av2HP shall share a common principle of using and re-assigning path indices to paths. In other words, they must share a common algorithm of conversion between absolute path identifier and path index. Please note that it is not necessary for the Av2HP and Av2HR to assign same absolute path identifier to the path, since this is only for internal referencing. The algorithm must just ensure that one path index designates the same 'real world' path on both sides of the communication channel.
In other words, when the sender intends to re-use a path index previously assigned to an absolute path (e.g. internally coded with Path ID #8880) and wants to associate a new path to it (e.g. with internal ID #9010), then the receiver must be informed about this fact so that the interpretation on both sides is the same – of course, the receiver may use different internal IDs than #8880 or #9010, as long as the same underlying topology is meant on both sides.

As a rule, in ADASIS v2 protocol, STUB message notifies client that new path exists in the ADASIS Horizon (section 4.6). Path Index of newly created path is contained in the field SUB-PATH INDEX of the STUB MESSAGE.

For example, the following stream of messages may flow from Av2 Horizon Provider to clients:

1. STUB message with SUB-PATH INDEX 8
2. SEGMENT message with PATH INDEX 8
3. SEGMENT message with PATH INDEX 8
4. STUB message with SUB-PATH INDEX 8
5. SEGMENT message with PATH INDEX 8

SEGMENT messages 2 and 3 refer to one path, while segment message 5 refers to another one, regardless of the fact that all three messages contain the same path index in the message. The presence of STUB message 4 marks the moment when path index 8 ceases to refer to the first path but starts to define a new one.

At this point one shall also realize that once a path index is 'recycled', there is no possibility for the Av2HP to send any new data for the first path. If this need arises (for instance, unexpected jump of position), then the 'old' path must be treated as a completely new one – a new index/identifier has to be assigned and all data for it would have to be re-transmitted under the new index.

One can implement the above approach on the client side in the following manner:

1. Client path identifiers internally use 32-bits and all path-related information uses this path identifier.
2. The low 6 bits of each path identifier are identical to the transmitted path indices; the high 26 bits is taken as the path's *generation*.
3. For each path index, 8...63, client keeps track of *current generation.*
4. When a message arrives, the internal path identifier to which the message refers is constructed by concatenating the 26-bit of current generation for a given index with index itself.
5. If the message refers to a STUB, then the generation number for the sub-path index is obtained by incrementing the current generation by one; this sub-path, as well as subsequent messages containing same message index, will then have this new generation (upper 26 bits), different from the previous one.

In the example before, the first two SEGMENT messages will be assigned to the path with identifier 72 (path index 8, generation 1). The second STUB message will increase the generation number, so that the third SEGMENT message belongs to the path with identifier 136 (path index 8, generation 2).

An advantage of this approach is that the client has more freedom in deciding which paths are not important anymore and may be discarded. In practical implementations, the client will still decide to keep only a limited number of paths, but then, this number depends only on available memory.

5.7.2 Offsets

If the ADAS sender or client needs an unlimited length of paths, this chapter describes a solution how to deal with this.

When having a potentially unlimited path length (however limited to $2^{32}-1$ meters for practical implementations), and having just 13 bit to transfer offset information, the obvious solution is to send the absolute offset modulo 8191. In other words, OFFSET m in Av2 messages in fact refer to an absolute offset of $x*8191+m$. The method to determine the implicit value of x can be solved using the following simple algorithm:

1. Client keeps track of offset generation x_{last}, and last received offset m_{last}.
2. In the start, $x_{last}=0$ and $m_{last}=0$.
3. After a message with OFFSET m is received:
    a. If $m >= m_{last}$, absolute offset is calculated as $x_{last}*8191+m$; and $m_{last} \leftarrow m$.
    b. For all messages except POSITION:
    if $m < m_{last}$, x_{last} is incremented by one, absolute offset is again $x_{last}*8191+m$ and $m_{last} \leftarrow m$.
    c. For POSITION messages:
    if $m < m_{last} - \Delta_{treshold}$, x_{last} is incremented by one, absolute offset is again $x_{last}*8191+m$ and $m_{last} \leftarrow m$.
    $\Delta_{treshold}$ is appropriate distance (say, 30 meters) that will allow back jumps of position along the path.

An Example of this calculation is given in Figure 19 and Table 25.

The client shall have one offset tracking pair of parameters x_{last} and m_{last} for each message type (POSITION, STUB …) and for each path index. Since there are 5 relevant message types (META-DATA do not contain OFFSET) and a maximum of 56 path indices, 280 tracking pairs are used in a maximum configuration.

In addition, Av2HP shall send, for instance, STUBs for one specific path sorted by distance – nearest STUBs first.

The same applies to PROFILE LONG and PROFILE SHORT messages. Profiles shall be sent by Av2HP sorted by distance.
If the client considers some profiles more important and if it wants to have profiles of one specific type along the path transmitted first, tracking of offsets must be performed for each profile type separately.

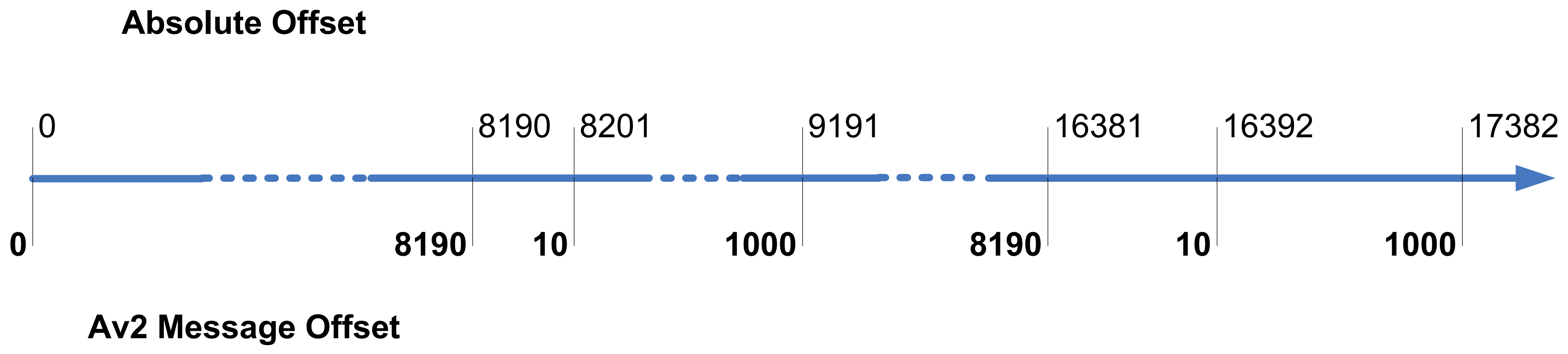


Figure 19: Message vs. absolute offsets

Table 25: Absolute offset calculation example

Message	x_{last} (before)	m_{last} (before)	MESSAGE offset m	x_{last} (after)	Absolute offset	m_{last} (after)
0	0	0				
1	0	0	8190	0	8190	8190
2	0	8190	10	1	8201	10
3	1	10	1000	1	9191	1000
4	1	1000	8190	1	16381	8190
5	1	8190	10	2	16392	10
6	2	10	1000	2	17392	1000
…	…	…	…	…	…	…

5.7.2.1 *Offset correction*

In some rare situations, events of some type along one path will be more than 8190 meters away. For instance, an application may be tailored only to work with speed cameras.

In that case, along one path the cameras may be positioned at absolute offsets 8190 and 16392. By using the offset tracking algorithm, offsets in messages will be 8190 and 10. Unfortunately, the absolute offset of the second camera will be then calculated to the closest 'alias', which corresponds to 8201. To correct the calculation, additional message need to be sent in order to force corresponding the x_{last} tracking parameter to be increased. The OFFSET field in that message must be smaller than the OFFSET contained in last message m_{last}, but greater than OFFSET that will be sent in the message containing the actual data. If necessary, multiple correction messages can be sent to force the x_{last} tracking parameter to be increased to the correct setting. Figure 20 illustrates this concept.

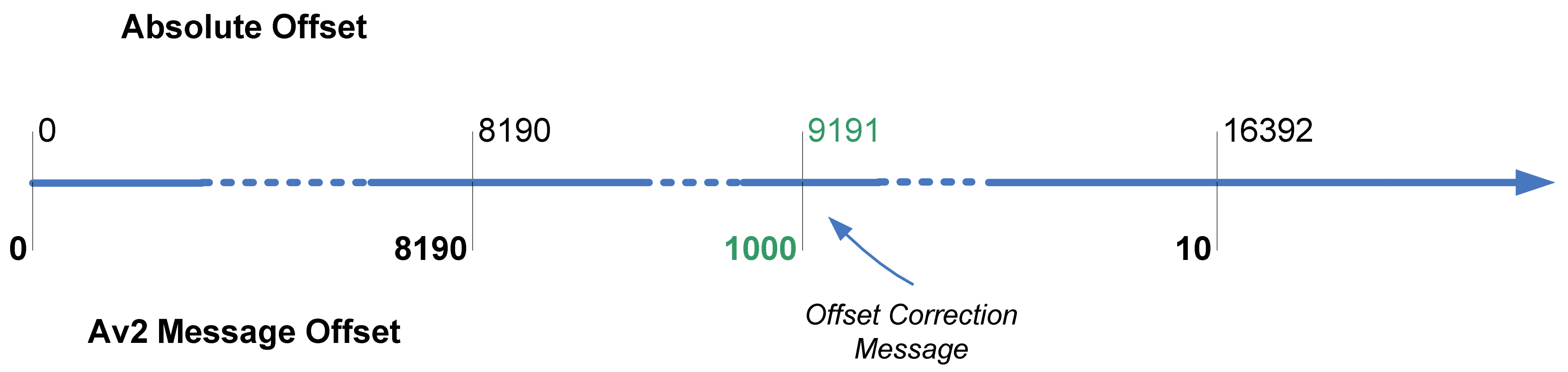


Figure 20: Offset Correction Message

For different message types, Offset Correction messages shall be realized differently:

- POSITION message do not need offset correction (section 5.5.1);

- STUB message offset correction is accomplished by sending continuation stub at appropriate OFFSET;
- SEGMENT offset correction is realized by sending exact content of last SEGMENT message with the same path, but with different OFFSET;
- PROFILE messages with invalid profile type can be used to correct offsets. As an alternative, if step or linear interpolated profiles are sent, the Av2HP can also send redundant messages instead. If, for instance, ROAD WIDTH profile is supported, and if road width between absolute offsets 8190 and 16392 is constant (say 8 meters), redundant message ROAD WIDTH PROFILE SHORT at absolute offset 9191 with value 8 will in effect correct offsets for all short profiles.
  A same approach is in principle possible when more sophisticated interpolations are used. On the other hand, this will require more complex calculations at the Av2HP side.

5.8 Profile Interpolation using Analytic Curves

In the ADASIS v2, *Profile* is a characteristic of a path that has a defined value for every location along the path. Typical examples of profiles are road curvature or road heading, but even absolute 2-D road geometry can be described as a profile (section 6.6).

5.8.1 Basic approach

Profiles are defined by specifying *profile support points* and by specifying the *interpolation* method that is to be used to calculate the profile between support points. The Interpolation method is defined by profile type (Table 17 and Table 22). In general, it is allowed that a single physical profile (say, curvature) is actually transferred using multiple profile types (say, CURVATURE/LINEAR and CURVATURE/STEP) that partitions the path into parts that are interpolated using, for instance, linear interpolation, and parts where step interpolation should be applied.

Profiles in ADASIS v2 protocol are communicated in a PROFILE messages (section 4.7 and 4.8). This message can be used in different ways to accommodate both simple as well as complex functions describing a particular profile. Even if a function that is more complex is used, the client can restrict itself to applying simpler interpolations if not enough CPU power or memory is available.

The System can communicate PROFILEs in a way to minimize traffic (Figure 21), or it can decide to send redundant data and to increase the robustness of the protocol (Figure 22). Based on received messages, the ADASIS v2 Horizon reconstructor can handle both variants at the same time.

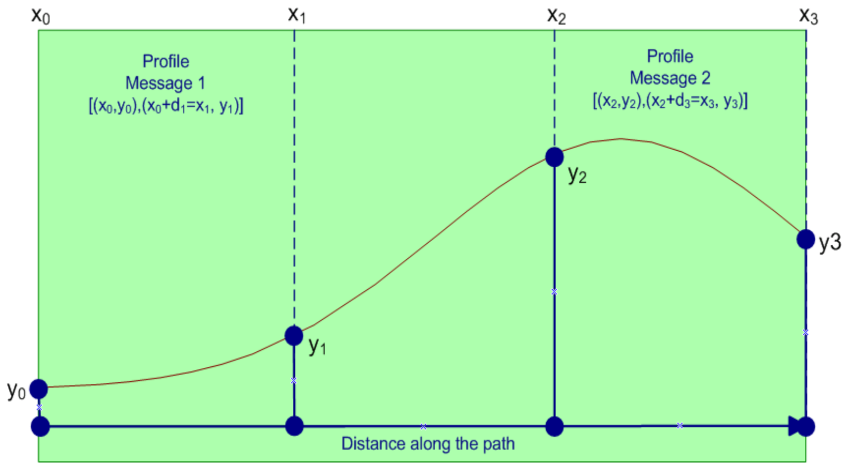

x0
x1
x2
x3
Profile Message 1 [(x0,y0),(x0+d1=x1, y1)]
Profile Message 2 [(x2,y2),(x2+d3=x3, y3)]
y0
y1
y2
y3
Distance along the path

Figure 21: PROFILEs in ADASIS v2 – optimized for bandwidth

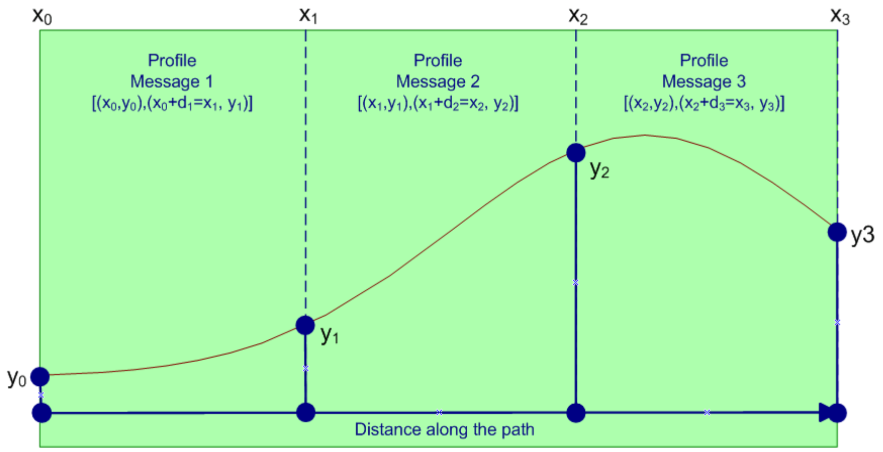

x0
x1
x2
x3
Profile Message 1 [(x0,y0),(x0+d1=x1, y1)]
Profile Message 2 [(x1,y1),(x1+d2=x2, y2)]
Profile Message 3 [(x2,y2),(x2+d3=x3, y3)]
y0
y1
y2
y3
Distance along the path

Figure 22: PROFILEs in ADASIS v2 – optimized for robustness

For ADASIS v2 Clients, the most commonly used interpolations of profile functions are *step* interpolation (Figure 23) and *linear* interpolation (Figure 24).

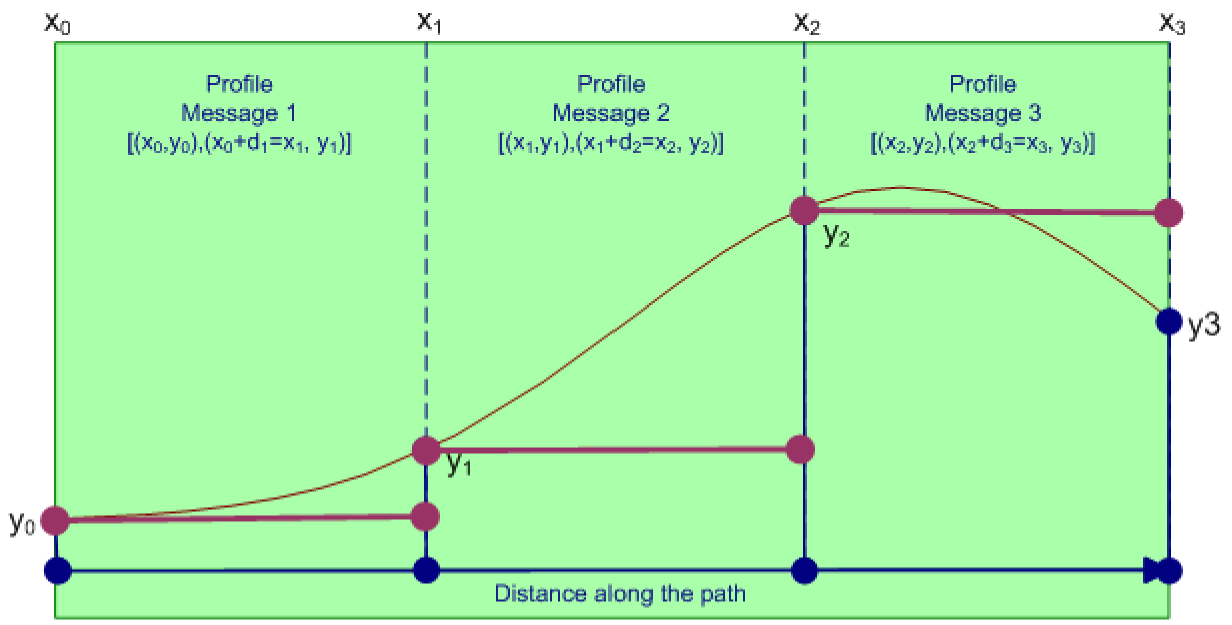

x_0
x_1
x_2
x_3
Profile Message 1 $[(x_0,y_0),(x_0+d_1=x_1, y_1)]$
Profile Message 2 $[(x_1,y_1),(x_1+d_2=x_2, y_2)]$
Profile Message 3 $[(x_2,y_2),(x_2+d_3=x_3, y_3)]$
y_2
y3
y_1
y_0
Distance along the path

Figure 23: Step Interpolation

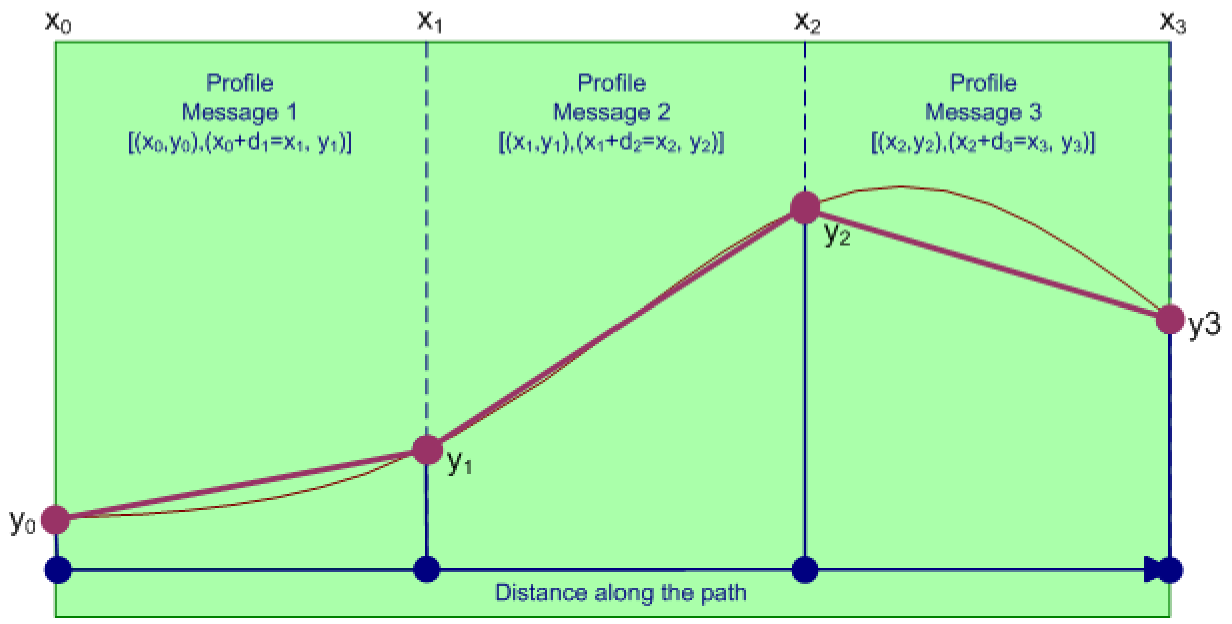

x_0
x_1
x_2
x_3
Profile Message 1 $[(x_0,y_0),(x_0+d_1=x_1, y_1)]$
Profile Message 2 $[(x_1,y_1),(x_1+d_2=x_2, y_2)]$
Profile Message 3 $[(x_2,y_2),(x_2+d_3=x_3, y_3)]$
y_2
y3
y_1
y_0
Distance along the path

Figure 24: Linear Interpolation

5.8.2 Complex interpolation functions

The CONTROL flag in the PROFILE message enables transfer of additional parameters that defines interpolation functions. For instance, step and linear interpolation are, generally speaking, polynomial interpolations of degree 0 and 1. By sending an additional PROFILE message with a *control* point, one can send an additional profile parameter that defines second-degree polynomial interpolation (Figure 25).

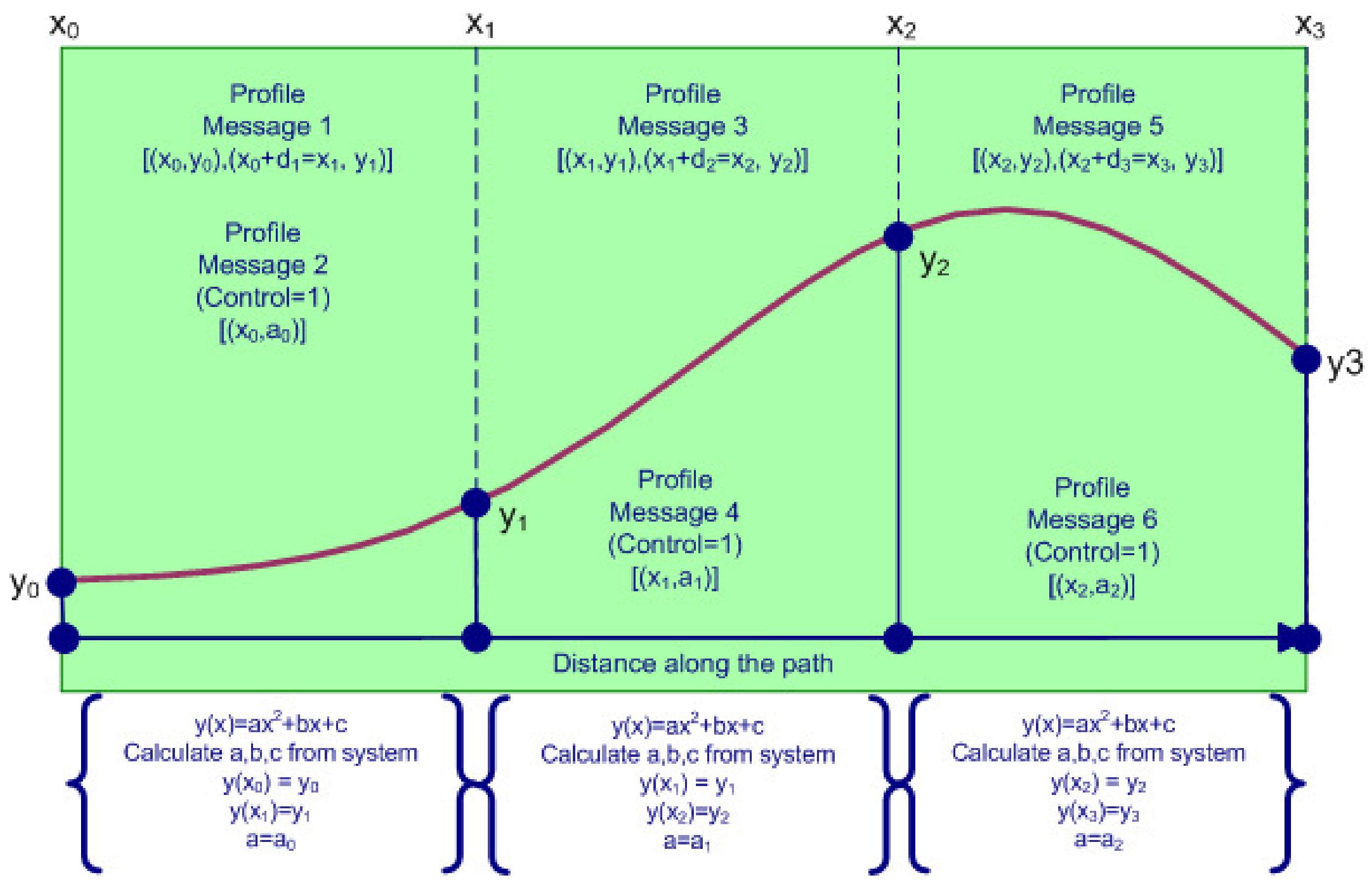

x_0
x_1
x_2
x_3
Profile Message 1 $[(x_0,y_0),(x_0+d_1=x_1, y_1)]$
Profile Message 2 (Control=1) $[(x_0,a_0)]$
Profile Message 3 $[(x_1,y_1),(x_1+d_2=x_2, y_2)]$
Profile Message 4 (Control=1) $[(x_1,a_1)]$
Profile Message 5 $[(x_2,y_2),(x_2+d_3=x_3, y_3)]$
Profile Message 6 (Control=1) $[(x_2,a_2)]$
y_0
y_1
y_2
y3
Distance along the path
$y(x)=ax^2+bx+c$ Calculate a,b,c from system $y(x_0) = y_0$ $y(x_1)=y_1$ $a=a_0$
$y(x)=ax^2+bx+c$ Calculate a,b,c from system $y(x_1) = y_1$ $y(x_2)=y_2$ $a=a_1$
$y(x)=ax^2+bx+c$ Calculate a,b,c from system $y(x_2) = y_2$ $y(x_3)=y_3$ $a=a_2$

Figure 25: Second-degree polynomial interpolations

The same mechanism can define a polynomial of arbitrary degree for the profile interpolation. For instance, to use a 5th degree polynomial, $y(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ the following PROFILE messages need to be sent.

Field	Value	Comment
Message Type	4	
...	**...**	**...**
Offset	x_0	Left boundary of interpolated interval
Control Point	false	
Long Value	false	
Value 0	y_0	$y_0=f(x_0)$, left boundary value.
Distance 1	$x_1 – x_0$	Length of interpolated interval
Value 1	y_1	$y_1=f(x_1)$, right boundary value.
...	**...**	**...**

Table 26: 5th-degree polynomial PROFILE interpolation – Message 1

Field	Value	Comment
Message Type	4	
...	**...**	**...**
Offset	x_0	Left boundary of interpolated interval identifies interval for which interpolation is defined.
Control Point	true	
Long Value	false	
Value 0	a_2	
Distance 1	*0*	Indicates 1st pair of interpolation parameters
Value 1	a_3	
...	**...**	**...**

Table 27: 5th-degree polynomial PROFILE interpolation – Message 2

Field	Value	Comment
Message Type	4	
...	**...**	**...**
Offset	x_0	Left boundary of interpolated interval identifies interval for which interpolation is defined.
Control Point	true	
Long Value	false	
Value 0	a_4	
Distance 1	*1*	Indicates 2nd pair of interpolation parameters
Value 1	a_5	
...	**...**	**...**

Table 28: 5th-degree polynomial PROFILE interpolation – Message 3

While a_2, a_3, a_4, and a_5 are explicitly defined, a_1 and a_0 needs to be calculated from boundary conditions defined in message 1.

This approach has the following advantages:

- The Client can interpret the profile as 1st order by simply ignoring CONTROL point messages; it can, consider only the first one and use linear interpolation.
- Missing parameters are easy to detect because the combination of OFFSET and DISTANCE 1 fields uniquely identifies the interpolation polynomial; the fallback solution is linear interpolation.

- The choice of interpolation functions is not limited to polynomials of up to 5th degree; every function whose parameters can be expressed as a combination of boundary conditions and explicit numbers can be used. If the calculation of parameters for out of boundary conditions is too complex, PROFILE/CONTROL messages will contain all parameters explicitly. It is still useful to send profile values on both ends of the interpolation interval in case the client wants to use linear interpolation.

5.9 ADASIS v2 Mini

Where clients are only interested in the attributes of the current link, Av2HP could decide not to send the ADAS Horizon in the form of Paths. Instead, a SEGMENT message with a PATH INDEX of 4 describes the attributes of the link where the vehicle is currently located. Since only one message is sent (or perhaps sent at regular intervals), we can refer to such an Av2HP as a minimal version of the system.

This minimal version is intended as an additional information layer to provide basic map information for the current vehicle position (e.g. to change the set-up of the air-conditioning in tunnels).

Since SEGMENT message of ADASIS v2 Mini is clearly different from “regular” SEGMENT messages, single Av2HP can support both regular ADASIS V2 and ADASIS v2 Mini in the same time.

6 Guidelines and Recommendations

6.1 ADAS v2 Horizon Provider (Av2HP)

Av2HP Recommendation #1 During system start, META-DATA messages should be transmitted more often to allow rapid start-up for clients requiring availability of basic configuration, map data, and ADAS v2 Horizon Provider characteristics.

Av2HP Recommendation #2 For ADAS applications that are sensitive to a loss of SEGMENT message in the middle of the path, LENGTH profile should be generated by the Provider. This PROFILE message will contain the offset at the start of the segment and length of the segment and enables the Av2HR to identify parts of the paths where values of attributes and profiles contained in the SEGMENT message are missing.

Av2HP Recommendation #3 The horizon provider should wait a certain distance until he re-uses path indices for new paths. If a path index e.g. 15 has already been used (not for the main path), the vehicle position should have passed the root of path 15 attached to the main path by at least the configured maximum back jump distance until this path index 15 can be re-used to attach a completely new path at the far end of the horizon. Also, the current vehicle position should be with a high probability not on a root with sub path index 15 until path index 15 can be re-used. This increases the robustness and prevents possible index mix-ups in the case of position jumps. For back jumps see also section 5.5, for position jumps see chapter 6.4.

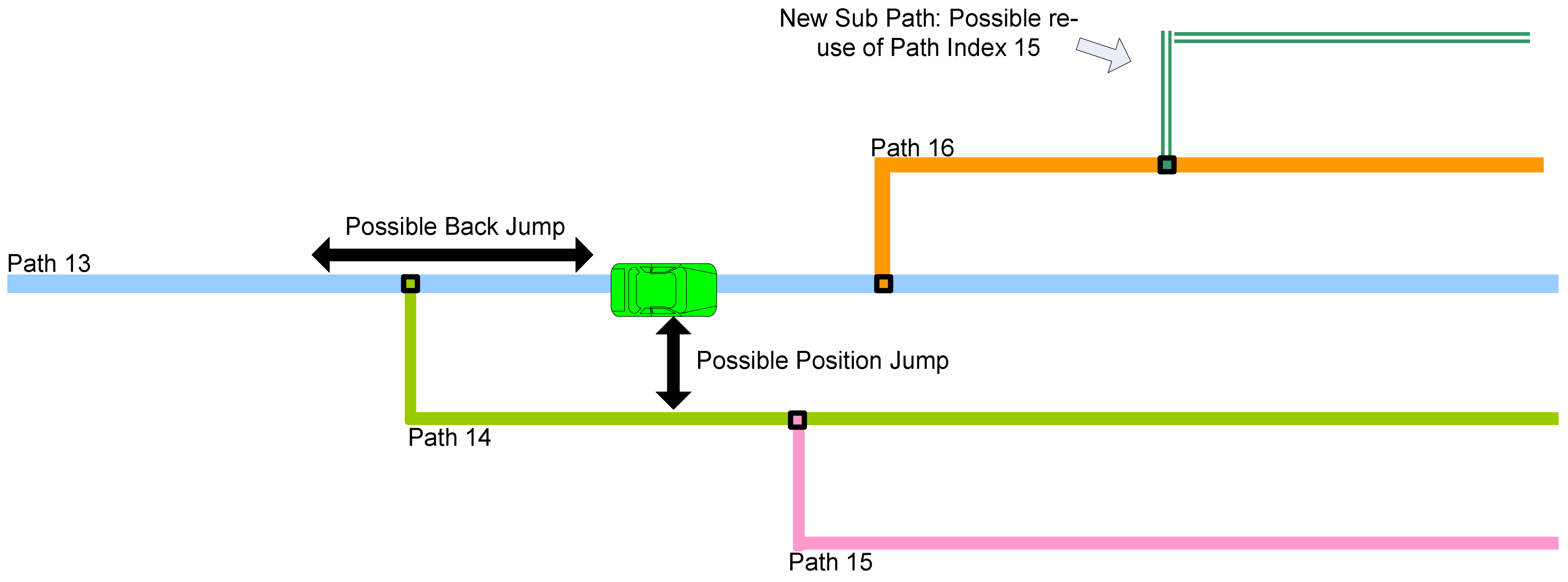


Figure 26: Re-Use of Path Indices

Av2HP Recommendation #4 The horizon provider should not use the complete maximum offset range. It should reduce the maximum path length of transmitted path by a certain distance to reduce the risk of mis-interpretations of cyclic offsets.

Av2HP Recommendation #5 It is recommended to start creating the horizon with the vehicle position located at some positive offset, because some applications might use the possibility to look backwards. A second reason is that especially at start-up; the current vehicle position is still inaccurate and could therefore jump backwards.

Av2HP Recommendation #6 If a certain attribute does not change for a long stretch of the horizon, the Horizon Provider should generate additional "no-change" messages to indicate that the information is still valid.
Example: If a vehicle follows a motorway, the map attributes in the SEGMENT message may not change for long distances e.g. 10 km. In the strict sense, it would be sufficient to send out this message only once. However, it would increase the reliability of the ADAS application if the Horizon Provider sends out a new message with the same content at regular intervals (e.g. every 1 km). Otherwise, it would not be possible for the Horizon Reconstructor to distinguish whether the map information is still valid or if the Horizon Provider has stopped sending out that information because of e.g. a software error.

6.2 Message Timings

System requirements will usually limit the number of messages that can be sent within a defined period, usually within one second. In addition, there should be a predefined minimum delay between messages. In this case, the Horizon Provider must handle these limitations.

While POSITION, SEGMENT, STUB, and PROFILE messages will be generated depending on the current vehicle position, the META-DATA message should be sent periodically. The time between two META-DATA messages, (for instance, one message every 5 seconds) is part of the configuration of the system.

It may be defined in the system requirements that the Horizon Provider should send exactly *N* messages per second (for instance, *N=10*). If there are no new messages that need to be sent, the Horizon Provider should choose and re-transmit one of the already sent messages with a certain priority scheme (e.g. randomly, sorted by distance, …). The retransmitted messages shall be marked with the “retransmission flag”= true and shall have exactly the same data content as the initially transmitted message (including the value of the cyclic counter).

In case no messages are available for re-transmission (for instance, at start of the system), the Horizon Provider may fill the message quota with META-DATA message.

6.3 POSITION message priority

Because of the importance of the position in applications, the lowest CAN id may be assigned to the POSITION message.

This may, however, in rare situations lead to inconsistent results at the reconstructed side.

Let us suppose that the POSITION message has a lower CAN id than the STUB message. Av2HP may generate messages in the following order:

1. STUB with path index 11;
2. STUB with path index 10;
3. POSITION with path index 10;
4. STUB with path index 11 (path index reused; old path with index 11 is obsolete);
5. POSITION with path index 11;

If messages are received in the same order, then the position will be correctly interpreted at all times:

a) No valid position prior to message 3,
b) Position at path with index 10 after messages 3 and 4;
c) Position at path with index 11 after message 5;

Because of higher priority of the POSITION over the STUB message, the order in which Av2HR receives messages may be:

1. POSITION with path index 10;
2. POSITION with path index 11;
3. STUB with path index 11;
4. STUB with path index 10;
5. STUB with path index 11.

From the Av2HR perspective:

a) Position is not valid after message 1;
b) Position is not valid after message 2;

c) Position refers to wrong path after message 3;
d) Position refers to wrong path after message 4;
e) Position finally refers to right path after message 5.

To achieve better consistency of positioning, POSITION message can have a higher CAN id than the STUB message. In that case, the order of receiving may be:

1. STUB with path index 11;
2. STUB with path index 10;
3. STUB with path index 11;
4. POSITION with path index 10;
5. POSITION with path index 11.

Now, the position will be either invalid (messages 1-3), or consistent (message 4 and 5).

This approach, however, may result in an unwanted latency of POSITION message.

6.4 Use of Alternative Positions in prevention of ADAS Horizon re-initialization

Within the Av2HP, more than one Map-Matched position might exist, especially in the case of a dense road network, where after passing an intersection it may not be clear which path the vehicle has taken. Especially if two alternatives have similar turn angles, such as at bifurcations, and run in parallel for a certain distance.

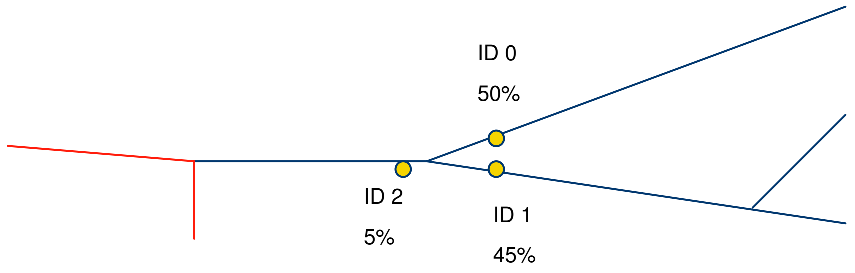
Since no explicit deletion message is defined for segments and stubs, the Av2HR needs to decide when to delete segments. This could be done for all data that is more than, for instance, 100m behind the vehicle and for the alternative paths that have origins on these deleted segments and therefore cannot be reached anymore. Such an approach has the disadvantage that segments associated with probable alternative positions, e.g. at bifurcations, can be deleted in the Av2HR. If the Av2HP position jumps to one of these alternative paths, it does not know if this new path is still known by the Av2HR or has already been deleted. It therefore needs to retransmit the data that may have been deleted and potentially the application will be “blind” during the transmission time. This retransmission would also increase the CAN traffic.

To overcome such an effect an option could be to keep the segments and alternatives behind the current position until the current position probability has exceeded a threshold of, for example, 90%. Unfortunately, in dense city networks this might never be the case, so deletion would not occur for a long time and the number of available segment indices would be exceeded.

For these situations, it is meaningful to transmit map matched position alternatives. If the sum of these position probabilities equals 100%, the Av2HR knows that it can delete all segments that are lying behind the addressed segments, as well as all their unreachable successors. The applications shall only use the first available position (index 0) as being the most probable alternative. The other position alternatives shall normally not be considered by the application. However, lower probability position alternatives may later turn to be the most probable one, so they are maintained by the Reconstructor for housekeeping of the path memory and coping with varying probabilities.

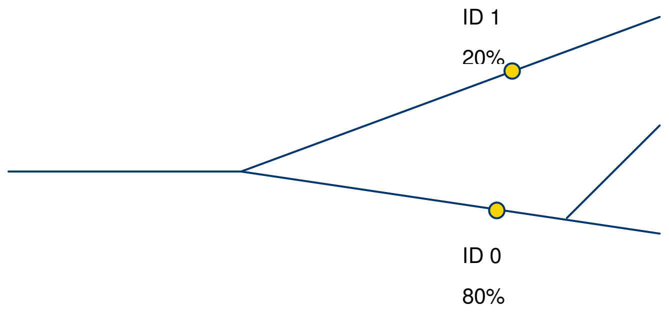
This approach avoids unnecessary retransmissions but requires a higher bandwidth since several position messages will be transmitted within one second instead of only one.

If future applications would need to know all (up to four) different positions at the same time, the specification provides for the possibility of using all alternatives.



Delete the red segments, because there is no position on them and they are out of the backward horizon.

ID 0 marks the Most Probable Path. The other positions are possible alternatives.



The Most Probable Path changed from the upper path to the lower path. Since that one was not deleted for having a position on it ,a retransmission is avoided.

Do not delete anything because all segments ahead can still be reached and the segment behind is still part of the backward horizon.

Delete the red segments, because there is no position on them, so, they and their successors are out of the backward horizon.

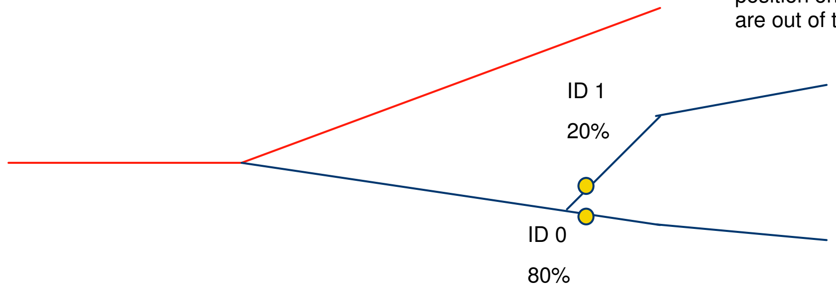


Figure 27: Alternative positions

6.5 Profile interpolation: pre-calculated data or real-time approximation

The function and its parameters that describe each profile along the path can be stored explicitly in the database. In this case, the task of the ADASIS v2 Horizon Provider is to read and format those parameters and send them to the Av2HP client.

Alternatively, profiles may be defined point-wise, for instance on each shape point.

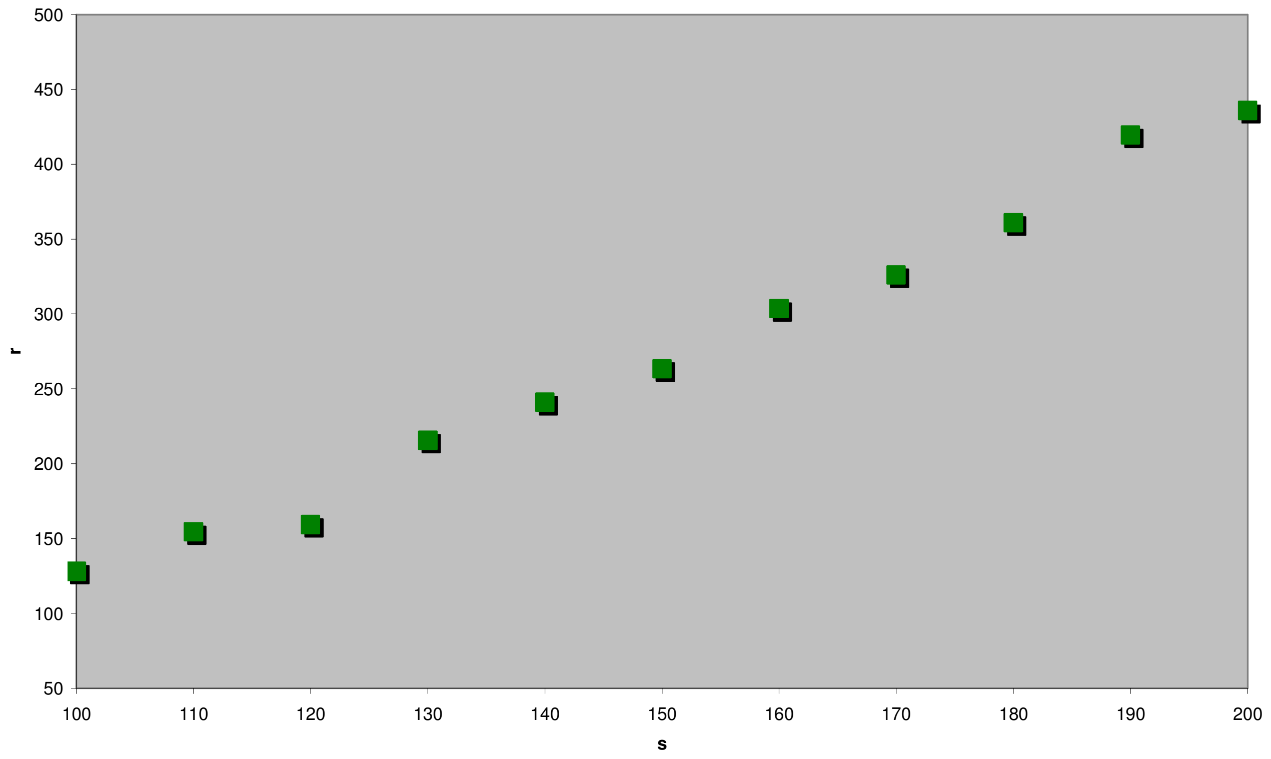


Figure 28: Shape-point based profile example (here: curvature radius r)

The straightforward implementation of the ADASIS v2 Horizon Provider will send one PROFILE message for each interval between shape points, and define step or linear interpolation for points in between.

Horizon Providers that are more sophisticated can try to find the best interpolation method in order to minimize network traffic.

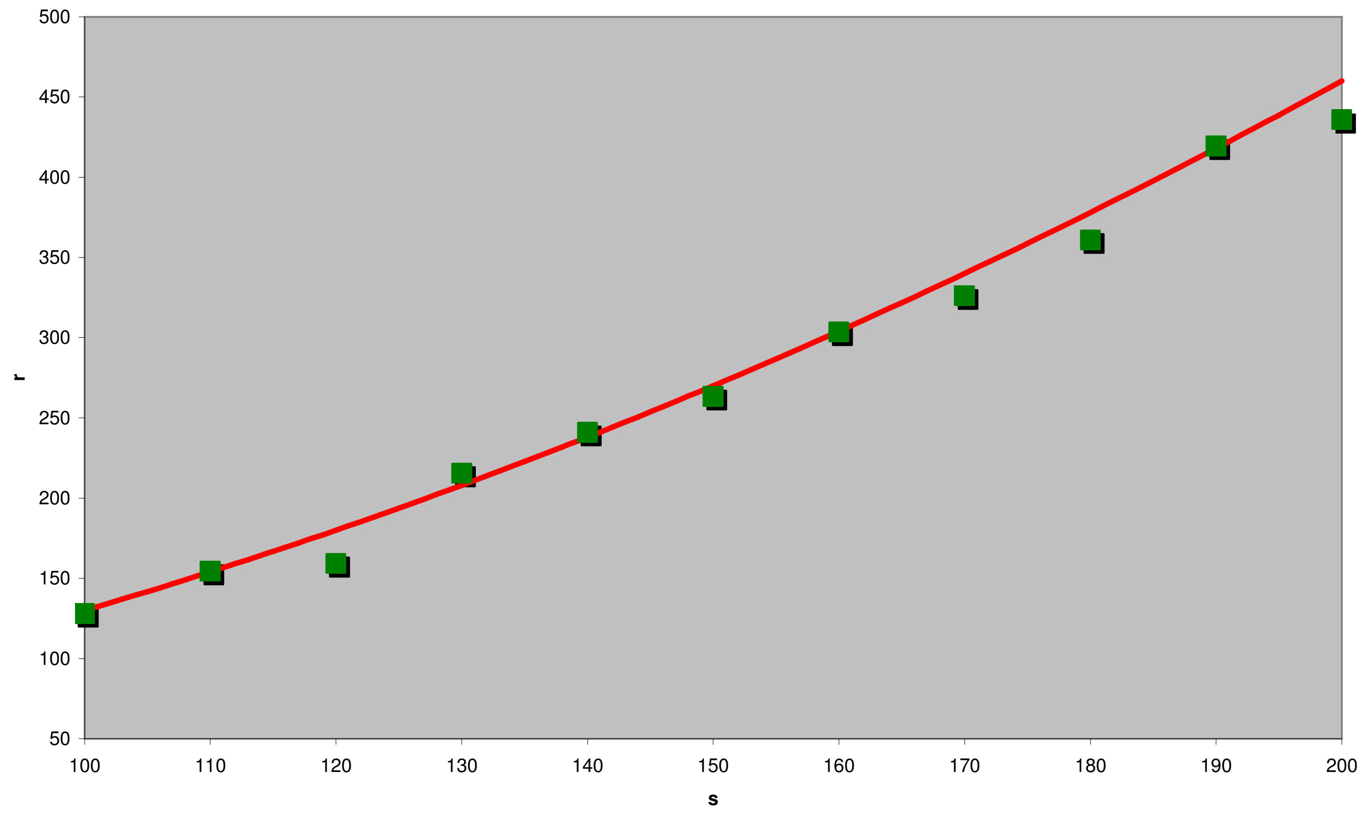


Figure 29: Optimized shape-point based profile

For instance, as shown in Figure 29, a 2nd degree polynomial may approximate data very well and the horizon provider can decide to send, instead of many short linearly-interpolated segments, one long interval on which a single, but more complex, interpolation can be applied.

6.6 Absolute Road Geometry

The absolute road geometry, expressed in terms of longitude/latitude coordinates has no explicit place in the ADASIS v2 protocol. We assume that in ADAS applications this information has very limited usability. Nevertheless, absolute road geometry can be easily published by ADASIS v2 Horizon Provider as two PROFILEs: LONGITUDE profile and LATITUDE profile.

For instance, when we want to transmit coordinates of shape points present in classical digital map databases, the following process can be implemented:

- Determine what shape points are along specific path;
- Calculate offsets of shape points along the specific path;
- For shape point at offset *x* on the specific path, generate signed 32-bit profile value of LONGITUDE;
- For shape point at offset *x* on the specific path, generate signed 32-bit profile value of LATITUDE.

Linear interpolation of LONGITUDE and LATITUDE PROFILEs will in effect result in classical, poly-line based road geometry. Other interpolation types can describe road geometry in ways that are more precise. For instance, second or third degree polynomial interpolations will represents roads as splines and this approach may reduce the number of shape points that need to be transmitted.

Alternatively, instead of sending 32-bit longitude/latitude values, the provider may send only 10-bit coordinate deltas relative to previous shape points. While the number of messages is reduced thereby, an additional PROFILE type must be created for sending origin points from time to time. In addition, such method is less robust because loss of message will invalidate remaining PROFILEs.

6.7 Sample system-specific PROFILEs

ADASIS v2 system can support up to 30 different 10-bit profile types (1-31). Of those, profiles with types 1-15 will always follow the standard as described in Table 22, while profiles 16-31 are system specific. Table 29 illustrates one possible definition of profile types.

PROFILE Type	*Description*	*Value Interpretation*	*Profile Range*	*Interpolation Type*	*Default Value*
0	Invalid				
1-15	See Table 17				
16	Banking	Super-elevation in 0.05 degrees	-25.55°... -25.55°	Linear	0
17	Banking (high resolution)	Super-elevation in 0.01 degrees	-5.11°... -5.11°	Linear	0
18	Lane 1+2 Width	Low 5-bit width of lane 1 in 0.25m units; high 5-bit width of lane 2 in 0.25m units	0 … 25.5 m per lane	Step	Country and road type dependent
19	Lane 3+4 Width	Low 5-bit width of lane 3 in 0.25m units; high 5-bit width of lane 4 in 0.25m units	0 … 25.5 m per lane	Step	Country and road type dependent

PROFILE Type	Description	Value Interpretation	Profile Range	Interpolation Type	Default Value
20	Lane 5+6 Width	Low 5-bit width of lane 5 in 0.25m units; high 5-bit width of lane 6 in 0.25m units	0 … 25.5 m per lane	Step	Country and road type dependent
21	Alternate radius	Curve radius in meters.			
22-31	Unused				

Table 29: System Specific PROFILE SHORT Types

6.8 Change of Speed Limit units in run-time

In rare occasions, Speed Limit units can change between kilometer per hour to miles per hour and vice versa during run-time. This will occur if the path is crossing a border between countries having different speed units.

META-DATA messages containing Speed Limit unit information are normally sent every several seconds. It is recommended to force transmission of a META-DATA message with changed speed units before the first SEGMENT message based on those units is generated

This solution is acceptable for cases where the META-DATA message and the SEGMENT messages share the same CAN identifier (multiplexed implementation). However, in non-multiplexed environments, a desired transmission sequence of SEGMENT messages in relation to the unit-defining META-DATA messages cannot be guaranteed. Speed Limit units of a SEGMENT message may then be wrongly interpreted by the reconstructor for a limited period after border crossing.

7 *Error Detection and Recovery*

The 2-bit Cyclic Counter that is part of each message can be used by the reconstructor to detect missing CAN frame; therefore ADASIS v2 protocol enables clients to detect up to 3 missing frames of each message type. The cyclic counter is not on the physical layer of the messages, but on the logical content layer. Each message type and each profile type has its own cyclic counter.

The ADASISv2 protocol provides the mechanism to repeat messages if there is free bandwidth. A re-sent message contains exactly the same content as the initial message except the special "retransmission" flag that is set to "true". If the horizon reconstructor has detected a missing frame, it can listen to the re-transmission messages in order to complete the message sequence.

This protocol is completely uni-directional and does not define messages that the client could send back to the ADASIS Horizon Provider to request the specific missing information. Depending on requirements, specific implementations shall implement custom error-recovery mechanism, such as message retransmission request: Av2HR sends to Av2HP message requesting retransmission of one or more messages or even re-initialization of complete ADASIS v2 Horizon.

One issue that is common to all types of missing messages is consistency of offset tracking (section 5.7.2). Simple way to make specific implementation robust in that respect is to limit length of the each path to 8190 meters (i.e. avoid using cyclic offsets). In that case, offsets contained in messages are absolute ones and one lost message will not influence consistency of offsets in next messages.

7.1 Missing POSITION message

Since in all configurations, POSITION message is periodically transmitted, the loss of one frame will only increase the position latency, as the client simply must wait for the next message.

7.2 Missing SEGMENT message

As consequence of SEGMENT message, Client will either have no attributes contained in the message about part of the path (in case of first segment of the path), or it will extend attributes from previous segment to the missing part. While first situation is relatively benign (application is aware of unknown data; it must be designed to handle such situation properly anyway), second case can present significant issue for some ADAS applications.

7.3 Missing STUB message

Missing STUB messages will pose a problem for applications for which knowledge about crossings is critical, since the corresponding road-branch at the intersection will be missing.

Moreover, since STUB messages actually inform the client about the creation of new sub-paths, the STUB entity is also used for later deletion of the path connected to it. Subsequent SEGMENTs and PROFILEs that refer to a missing STUB sub-path identifier may therefore be wrongly associated to the wrong path.

7.4 Missing PROFILE message

Since PROFILE messages are part of the path to which they apply, the application can easily recognize where PROFILE data are not or only partly available and deal with this situation. For critical applications, redundant PROFILEs may be generated and transmitted as well.

In case of a missing PROFILE message containing CONTROL points, the client can choose the fallback to linear interpolation of the profile on that part of the path.

7.5 Missing META-DATA a message

META-DATA messages are sent in periods of several seconds, but contain data that rarely change. Therefore, loss of this message type is not critical for the client function – he simply will catch up when receiving the subsequent META_DATA message. An exception is at the very start of the system, during which the client should rapidly pick up some configuration data from that message. To prevent problems, it is recommended, to transmit META-DATA messages at higher rate after system start-up.

8 *Examples of use*

8.1 Different ADAS Horizon Requirements in the Vehicle Network

The typical vehicle set-up has one ADAS Horizon Provider and may contain multiple clients for different applications. The applications may differ in many ways:

- Different map information requirements
- Different control units with different storage capacity and processing power
- Different bus connection bandwidth

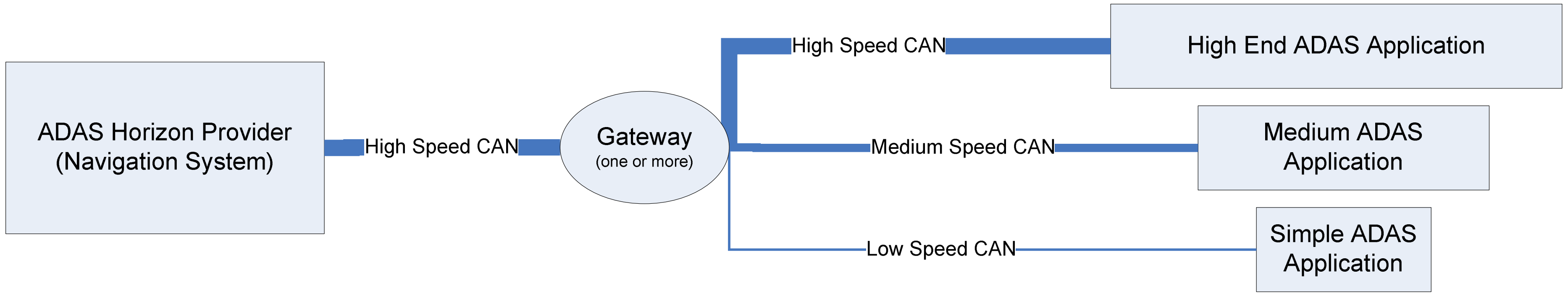


Figure 30: Possible Vehicle Network with different ADAS Applications

The heterogeneous applications listening to one ADAS horizon broadcast stream require compromises in the parameter settings of the ADAS Horizon Provider to fulfil the different requirements as well as possible.

The different combination of clients in each vehicle will lead to many different possible configuration variants of the ADAS horizon provider. The following section describes two examples and solutions with the ADASIS v2 protocol.

8.1.1 Example Set-up 1: Cyclic offsets with different path lengths

One possible requirement could be that the simple ADAS application can only store a 1000 m ADAS horizon due to storage capacity constraints whereas the high-end ADAS application requires an ADAS horizon of 7 km. The ADAS horizon provider can use the message retransmission concept to fulfil both requirements.

At system start-up, the ADAS Horizon Provider will transmit the complete 7 km, but the simple application will only store 1 km (see Figure 31).

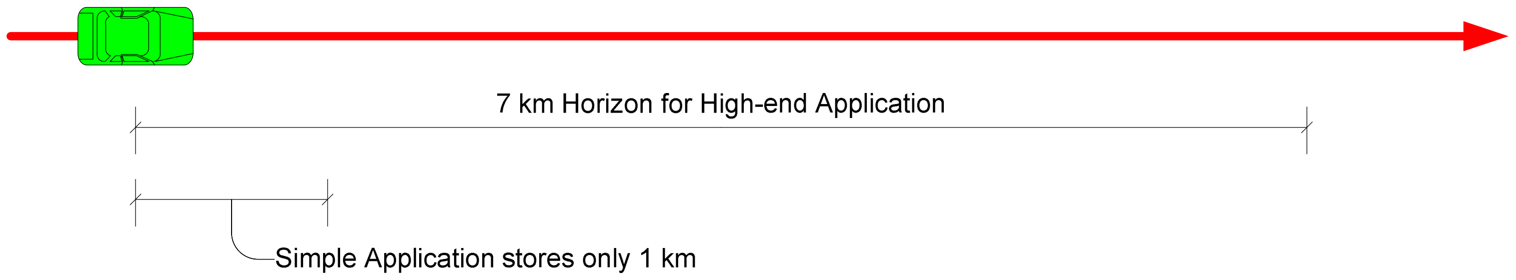

7 km Horizon for High-end Application
Simple Application stores only 1 km

Figure 31: Different Horizon lengths after start-up

As soon as the maximum length of the transmitted path has been reached, the Horizon Provider needs to follow both lengths (7 km and 1 km) internally. As the vehicle moves forward, the Horizon Provider will attach new information at the end of the 7 km horizon and retransmit already sent information at the end of the 1 km horizon (see Figure 32). The high-end application can filter out the retransmitted information immediately and the simple application discards all information with a distance to the vehicle > 1km.

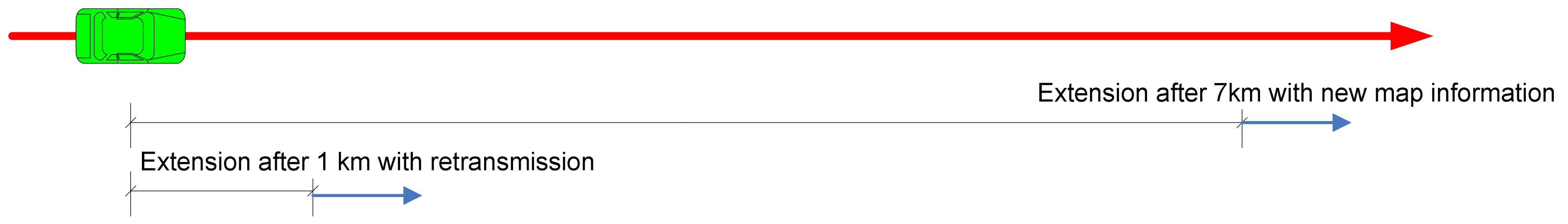

Extension after 7km with new map information
Extension after 1 km with retransmission

Figure 32: Extension of short Horizon with Message Retransmission

8.1.2 Example set-up 2: Multi-path and main path only

Another possible requirement could be that a simple application can only store information for 500m of the main path and that a more sophisticated application requires a transmitted path length of 5 km with stubs and sub-paths.

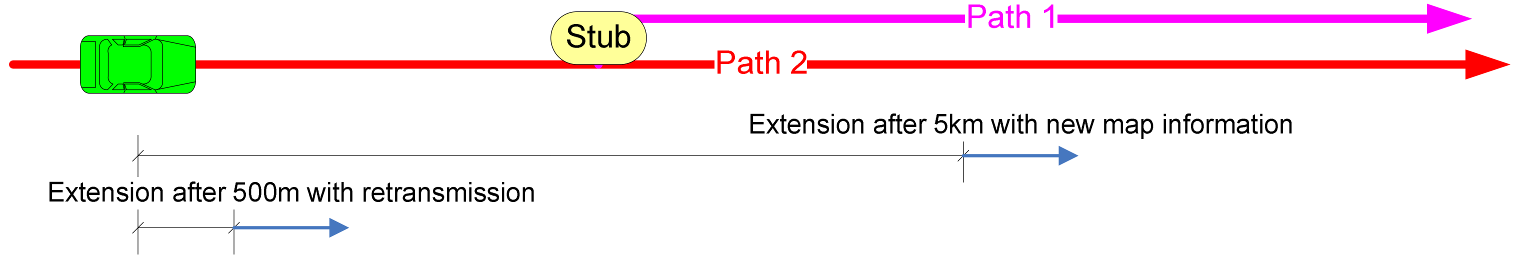

Stub
Path 1
Path 2
Extension after 5km with new map information
Extension after 500m with retransmission

Figure 33: Message Retransmission for Sub Paths

In this case, the Horizon Provider would need to attach new information to the long 5 km horizon and at the same time retransmit information for the short 500m horizon (as described in section 8.1.1). As an additional requirement, the Av2HP needs to retransmit the complete information of the sub path (Path 1).

If the vehicle drives from Path 2 to Path 1, the high end application would be able to continue without any "blind spot" in the map information. As soon as the vehicle position changes from Path 2 to Path 1, the simple application would have to discard all horizon information, accept a short gap in the horizon and wait for retransmission messages for the beginning of Path 1.

8.2 Example Applications and possible Configurations

There are several questions with respect to client requirements a system architect must answer during the definition of an Av2-Horizon configuration:

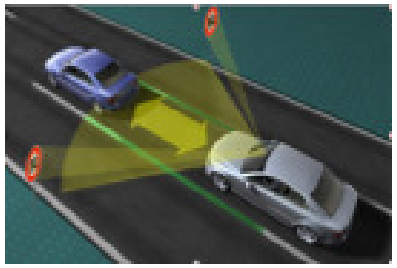
1. Can the applications accept lack of data for a few seconds? If yes, the single-path horizon can be used. If not, is it acceptable to send just 1st-level sub-paths, by which 'preview blindness' situations are strongly reduced, but not eliminated. In the ultimate case when all sub-path information is required to be present on the client-side, a multi level sub-path scheme must be chosen.
2. Do applications need crossing information? If yes, STUBs must be sent. Depending on the answer to the first question, one must choose between single-path horizon with stubs only or single-path horizon with stubs and 1st-level sub-paths etc.
3. Do applications rely on attributes of the SEGMENT message? If not, SEGMENT messages need not to be sent.
4. Which additional attributes are of interest? If SEGMENT and STUB messages contain all necessary data, PROFILEs are not needed.
5. Is it more efficient for clients to track offset overflows (section 5.7.2), or it is more efficient to concatenate paths? In the former case, unlimited paths can be used; the latter implies the use of paths with limited length.

As we will see below, besides POSITION message, in many configurations Av2HP needs to send just one message type in addition – STUBs, SEGMENTs or PROFILEs – but there may be no need to send anything else.

However, if there are multiple Av2 clients, the provider must send the total combination of all data required by any of them. Even the less demanding clients must be able to handle all received data in a consistent way, i.e. either process or ignore the sent data.

For instance, a single-path horizon application must be also able to track STUB messages in case the provider is configured to send them (e.g. for another client connected in the same environment).

8.2.1 Speed Limit Warning



Speed Limit Warning is the simplest form of an application, since it is not safety-critical. Here, the unlimited single-path horizon set up would be sufficient. The Av2HP can send only SEGMENT messages and POSITION message to satisfy all requirements. Paths will be implicitly created on the client side, when new SEGMENT message do not contain same PATH INDEX as previous one (section 5.7.1). Data fusion with traffic sign recognition may bring advantages of combining up to date information from the camera with implicit speed restrictions provided by the navigation system.

8.2.2 *Curve Warning*

This application may be seen as safety-critical, but in most cases a single-path horizon with 1st-level sub-paths may serve. SEGMENT messages do not contain any information of interest, so only POSITION, STUBs (for notification about new paths) and PROFILE SHORT messages (for CURVATURE profile) are needed.

8.2.3 *Intersection Warning*

Since this application is focused on intersections only, no SEGMENT or PROFILE information are of general use. Av2HP need to send only POSITION and STUB messages.

8.2.4 *Lane Change Assist*

This application provides warnings for approaching the target lane in combination with a blind spot detection sensor. To enhance the function, additional warning with respect to road information (no passing signs, lane numbers etc.) could be given. This application could work with POSITION, SEGMENT and PROFILE messages only, although the system would perform better in full horizon mode having alternative paths available.

8.2.5 *Vehicle Dynamics Adjustment*

The driver chooses an operation mode (economy, normal, sport) in which adjustments of several dynamic settings are adapted, e.g. damper control, pre-fill of brakes, or even other ADAS applications like curve warning. The ADASIS v2 Horizon could support this function by curve data, road condition, road type or slopes. For this kind of application the Av2HP must send the full set of message types.

8.2.6 Fuel Saving Application

Based on the slope profile of the road ahead, potential stop points and road geometry, not only eco driving assistance but also real fuel saving applications can be realized. Especially for hybrid energy management the eHorizon sensor enables a high efficient usage of battery. The Av2HP sends only POSITION and PROFILE information, potential stop points can be compiled on the provider side so that there is no need for STUB messages.

8.3 Clothoid Profiles

Clothoids (Figure 34) are 2-D parametric curves with the property that their curvature is a linear function of its arc length. The Clothoid is also known as a Cornu spiral.

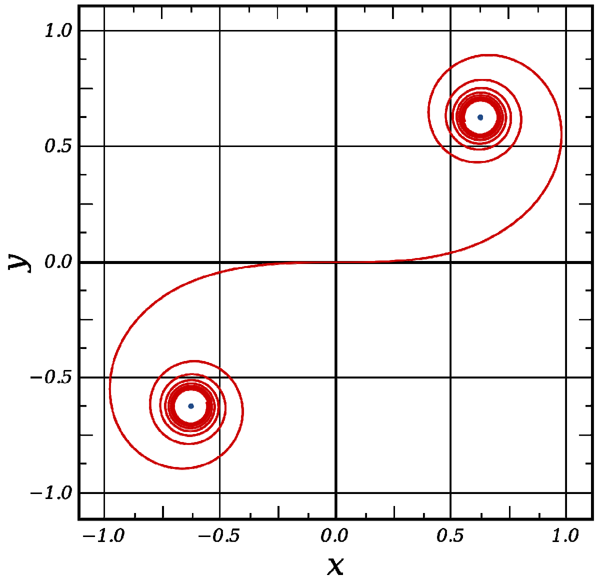


Figure 34: Unit clothoid

In some systems, Clothoids are used for the modelling of road geometry instead of the classical poly-line approach. In this case, parts of the Clothoids are used to model parts of the road. In theory, an ideal curve consists of a straight part, a transition curve and a circular arc (Figure 35). Clothoids are ideal transition curves since their curvature is linearly increased along the clothoid.

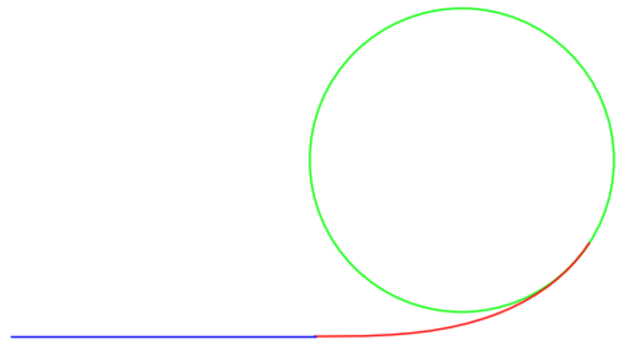


Figure 35: Straight (blue); transition curve (clothoid, red); circular arc (green)

Unfortunately, the majority of roads are not built according to this ideal and in practice one needs a number of Clothoids to represent a single stretch of road.

8.3.1 Clothoid for Relative Geometry

If absolute road geometry (longitudes and latitudes) are not of interest to the application, relative clothoidal geometry of a single path can be realized only by using the standard curvature profile (see section 9.1). Since clothoid *are* defined as curves for which the curvature is a linear function of distance, this approach will perfectly describe the target clothoid. One must be aware, however, that in this case OFFSETs must also be expressed as real distance between points, rather than Euclidian distance between two 2-D points. If necessary, additional, system-specific profiles containing arc distances between offsets may be introduced.

Another issue can be the continuity of Clothoids when the previous one does not end where the next starts. If the end curvature of the first clothoid is equal to the start curvature of the next one, no additional work needs to be done. Otherwise, two profiles need to be used: one containing the clothoid start curvatures, and another with the clothoid end curvatures. This approach will allow for discontinuities in curvature profiles.

8.3.2 Clothoid for Absolute Geometry

Each Clothoid segment can be represented by the following parameters (Figure 36):

- End points $P_s(lat, long), P_E(lat, long)$:
- End curvatures κ_s, κ_E or end heading angles θ_s, θ_E
- Length L

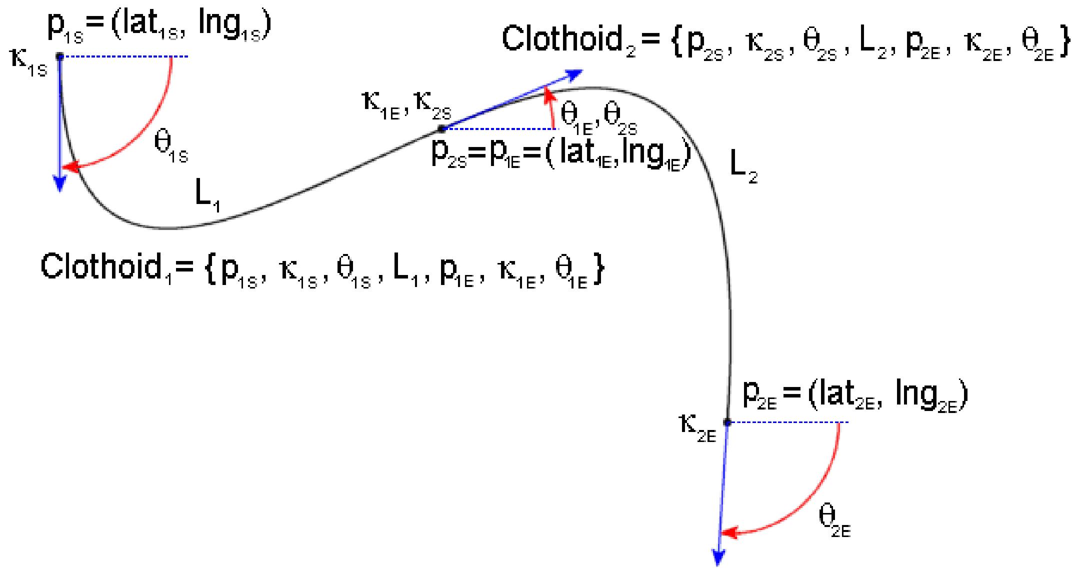

$p_{1S} = (lat_{1S}, lng_{1S})$
κ_{1S}
θ_{1S}
L_1
κ_{1E}, κ_{2S}
θ_{1E}, θ_{2S}
$p_{2S} = p_{1E} = (lat_{1E}, lng_{1E})$
$Clothoid_2 = \{p_{2S}, \kappa_{2S}, \theta_{2S}, L_2, p_{2E}, \kappa_{2E}, \theta_{2E}\}$
L_2
$Clothoid_1 = \{p_{1S}, \kappa_{1S}, \theta_{1S}, L_1, p_{1E}, \kappa_{1E}, \theta_{1E}\}$
$p_{2E} = (lat_{2E}, lng_{2E})$
κ_{2E}
θ_{2E}

Figure 36: Clothoid Representation of Roads

Therefore, in addition to standard LONGITUDE/LATITUDE profiles, one needs to transfer three additional parameters: (κ_s, κ_E, L) or (θ_s, θ_E, L). It will not usually be possible to achieve C^2 continuity so κ_s of the second clothoid will not be equal to κ_E of the first one. In other words, five numbers will be necessary to describe one part of the road using the clothoid model (assuming that one clothoid ends where the next one starts).

A consistent method of transferring clothoid data using this protocol is:

- Use of standard LONGITUDE/LATITUDE profiles;
- Define additional CLOTHOID PROFILE LONG. The spot value of this profile is clothoid length L, but an additional two control points are required: κ_s, κ_E or θ_s, θ_E.

8.4 Cubic Bézier Spline Profile

For some systems, Poly-line geometry of roads defined by shape points with Longitude and Latitude Profiles is sufficient. For others, resolution, range and overall characteristics of Curvature and/or Slope profiles may not be satisfactory. In that case, absolute and relative road geometry can be described using continuous functions such as Clothoids (section 8.3), B-splines or Cubic Bézier Splines.

Some map providers offer an alternative road geometry definition in form of Cubic Bézier Splines. To transfer those data using ADASIS v2 protocol, PROFILE LONG of types 1, 2, 3, 4, 5 and 6 are used to define a Cubic Bézier Spline Profile.

A Bézier curve is defined by its control points P_i:

$$B(t) = \sum_{i=0}^{n} P_i b_{i,n}(t) \quad where \quad b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0,\ldots n$$

where $b_{i,n}(t)$ are the Bernstein polynomials.

A Cubic Bézier curve is defined by its 4 control points [P_0, P_1, P_2, P_3]:

$$B(t) = \sum_{i=0}^{3} P_i b_{i,n}(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3, \quad t \in [0,1]$$

In case of a digital map, each control point is actually triple consisting of Longitude, Latitude and Altitude.

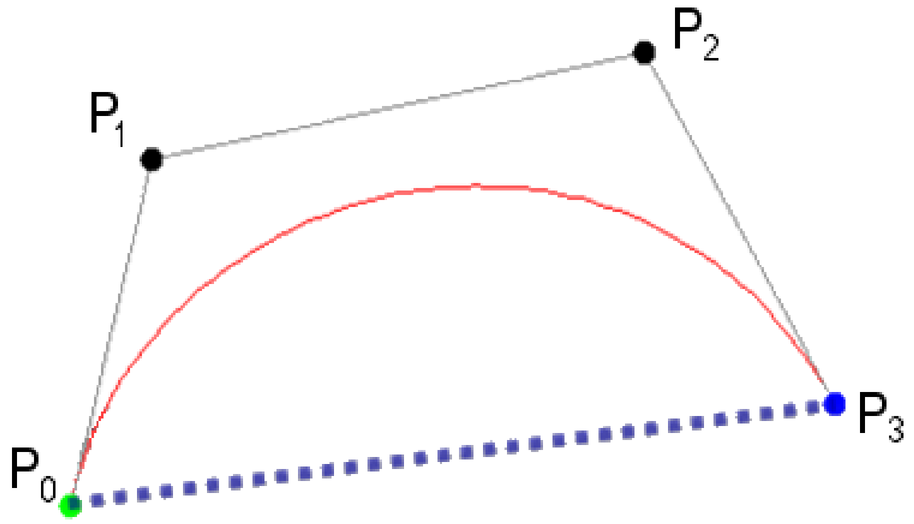


Figure 37 Cubic Bézier Spline (red) and its control points

As we can see from Figure 37, control points P_0 and P_3 are indeed standard geometry shape points that can be transferred using usual Longitude, Latitude and Altitude Profiles (PROFILE LONG types 1,2 and 3). To transfer P_1 and P_2 one need to use (Bézier) Control Point Longitude, Latitude and Altitude Profiles (PROFILE LONG types 4, 5 and 6).

- P_0 longitude, latitude and altitude PROFILE LONG messages are characterized by same offset X_0;
- P_3 longitude, latitude and altitude PROFILE LONG messages are characterized by same offset X_3;
- P_1 Bézier Control Point longitude, latitude and altitude PROFILE LONG messages are characterized by same offset X_1;
- P_2 Bézier Control Point longitude, latitude and altitude PROFILE LONG messages are characterized by same offset X_2;
- It must be $X_0 < X_1 < X_2 < X_3$. In ideal case, difference $X_3 - X_0$ is equal to the length of the Bézier curve arc. Offsets X_1 and X_2 do not have other meaning or purpose except as indication of order of Control Points.

If ADASIS v2 Horizon Providers publishes all above values, a client can choose

- To use P_0 and P_3 longitude and latitudes, getting classical 2-D poly-line geometry;
- To use P_0 and P_3 longitude, latitudes and altitudes, getting classical 3-D poly-line geometry;
- To use P_0 and P_3 longitude and latitudes together with P_1 and P_2 control longitudes and control latitudes describing high-resolution 2-D Cubic Bézier road geometry;
- To use all data that defines full 3-D Cubic Bézier road geometry.

In the same manner, road geometry described in form of Quadratic Bézier splines can be represented in the ADASIS v2 Protocol. In that case, only a single Bézier Control Point P_1 between end points P_0 and P_3 is defined. Bézier splines of higher orders are defined by having more than two Bézier Control points.

Please note that a single path will consist of several Bézier splines. Since the first point of the next spline is the same as the last point of the previous one, it needs to be transferred only once.

8.5 Link Identifier Profile

A Link Identifier Profile is used to uniquely identify parts of the path during multiple executions of the application. Besides the possibility to identify common segments of different paths, this feature may be used by the application to permanently store data related to specific roads. For instance, the application may need to store information associated to the link with identifier 34567. This information will be available for all current and subsequent paths that contain the link in question (see Figure 38).

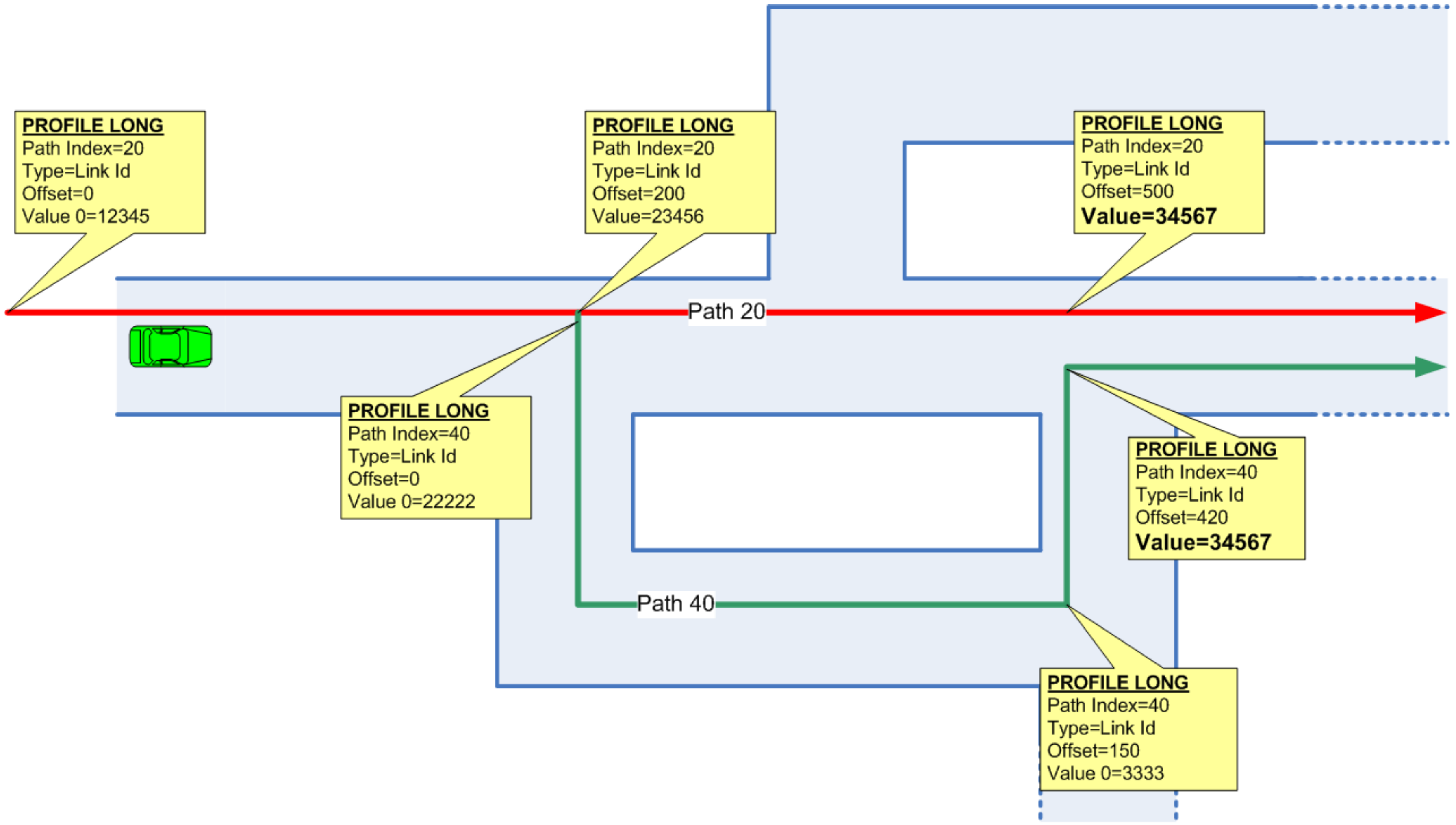


Figure 38 Permanent Link Id Profile

Internally to ADASIS v2 Horizon Provider, this profile may correspond to the database links that build up a single path.

GDF, which is a standard interchange format for map data, includes an object feature called *Published Version Identifier* (PVID). For the database links, we can speak of a *Link Identifier*. This identifier is unique in a single map and does not change between map releases.

The GDF PVID field can theoretically hold identifiers in a range between 0 and 9,999,999,999 (0x2540BE3FF), which is outside of range of 0 to 4,294,967,294 (0xFFFFFFFE) that fits into the dedicated PROFILE LONG message.

Currently (status December 2010), for example the NAVTEQ Map maximum PVID world-wide is around 900,000,000 – well within 32-bit identifier range. Taking into account the current rate of map coverage expansion and change rate one can expect that 32-bit range will be exhausted in few years.

In other words, contemporary systems can use GDF PVID as source of data for the Link Identifier Profile. To achieve uniqueness, future ADASIS v2 Horizon Providers may use combination of Link Identifier Profile and the Country Code from META-DATA message. Alternatively, new PROFILE LONG type may be defined that will contain additional 32-bit of the Identifier. Of course, those two PROFILE LONG will contain same Path Index and same Offset.

Map Databases that do not include GDF PVID will take some other identifier as ADASIS v2 Link Identifier.

9 Appendix – Value Tables

9.1 Curvature Interpretation

The curvature profile is modelled as a piecewise linear function. The curvature values changes by fixed steps which are constant within in 16 consecutive blocks of 64 bit span each (see coloured blocks in Table 30).

High curvature values for sharp curves (i.e. small radius) are coded with a low resolution whereas low curvature values (i.e. big radius) are coded with a high resolution. The ΔR value in the following tables illustrates the effect on resolution of the radius.

The coding is based on simple functions with power of 2 factors to allow easy calculation also on simple processing units without floating point calculation capability.

The piecewise linear functions are defined by a set of simple systematic formula. Negative values shall be mirrored by the curvature value 0. Positive values represent right curves, negative values left curves. Left curves with curvature<-0.16192 shall be coded with value 0, right curves with curvature >0.16192 shall be coded with value 1022. Value 1023 means "unknown".

Some values at the very high and very low end of the radius scale are not very meaningful from application point of view, but are defined in the table to follow the regular coding scheme.

Table 30: Curvature Interpretation (10-bit)

Message Value	*Δc [1/m]*	*Curvature c [1/m]*	*Radius R [m]*	*ΔR [m]*
0…510	*negative value range symmetrical to range 512 - 1022*			
511	0.00001	0.00000	∞	∞

512		0.00001	100000.000	50000.000
513		0.00002	50000.000	16666.667
...			…	
573		0.00062	1612.903	25.602
574		0.00063	1587.302	24.802
575		0.00064	1562.500	47.348
576		0.00066	1515.152	44.563
...	0.00002			
637		0.00188	531.915	5.599
638		0.00190	526.316	5.482
639		0.00192	520.833	10.629
640		0.00196	510.204	10.204
...	0.00004		…	
701		0.00440	227.273	2.048
702		0.00444	225.225	2.011
703		0.00448	223.214	3.916
704		0.00456	219.298	3.781
…	0.00008		…	
765		0.00944	105.932	0.890
766		0.00952	105.042	0.875
767		0.00960	104.167	1.708
768		0.00976	102.459	1.653
…	0.00016		…	
829		0.01952	51.230	0.417
830		0.01968	50.813	0.410
831		0.01984	50.403	0.800
832		0.02016	49.603	0.775
…	0.00032		…	
893		0.03968	25.202	0.202
894		0.04000	25.000	0.198
895	0.00064	0.04032	24.802	0.388
896		0.04096	24.414	0.376
…			…	
957		0.08000	12.500	0.099

958		0.08064	12.401	0.098
959		0.08128	12.303	0.191
960		0.08256	12.112	0.185
…	0.00128		…	
1021		0.16064	6.225	0.049
1022		> 0.16192	< 6.176	
1023	-	unknown	unknown	-

The coding based on curvature values *c* leads to an Integer representation having a resolution of 10^{-5} [1/m]. All coding and decoding calculation involves only add, subtract or bit-shift operations. Especially decoding can therefore be handled by simple controller units not having an integrated Floating-Point-Unit.

9.1.1 Coding method:

Notes:

- *c* is the *curvature* value in 1/m units to be represented
- *value* is the coded curvature, represented by a 10 bit integer value
- |*x*| represents the absolute value of *x*
- SIGN (*c*) represents the value -1 or +1 when *curvature < 0 or ≥0* , respectively
- ROUND (x) represents the (signed) integer closest to x

If $|c| < 0.00064$
    then *value* = $511 + \text{ROUND}(c * 100000)$

if $0.00064 \le |c| < 0.00192$
    then *value* = $511 + \text{SIGN}(c) * 32 + \text{ROUND}((c * 100000) / 2)$

if $0.00192 \le |c| < 0.00448$
    then *value* = $511 + \text{SIGN}(c) * 80 + \text{ROUND}((c * 100000) / 4)$

```
if $0.00448 \le |c| < 0.00960$
    then $value = 511 + \text{SIGN}(c) * 136 + \text{ROUND}((c * 100000) / 8)$
if $0.00960 \le |c| < 0.01984$
    then $value = 511 + \text{SIGN}(c) * 196 + \text{ROUND}((c * 100000) / 16)$
if $0.01984 \le |c| < 0.04032$
    then $value = 511 + \text{SIGN}(c) * 258 + \text{ROUND}((c * 100000) / 32)$
if $0.04032 \le |c| < 0.08128$
    then $value = 511 + \text{SIGN}(c) * 321 + \text{ROUND}((c * 100000) / 64)$
if $0.08128 \le |c| < 0.16192$
    then $value = 511 + \text{SIGN}(c) * 384 + \text{ROUND}((c * 100000) / 128)$
if $0.16192 \le |c|$
    then $value = 511 + \text{SIGN}(c) * 511$
```

9.1.2 Decoding method

Notes:

- *value* is the 10-bit coded curvature value
- a *value* of 1023 signifies 'curvature unknown'
- $C = value - 511$ (signed Integer)
- SIGN (C) represents the value -1 or +1 when $C < 0$ or $C \ge 0$, respectively

```
if $|C| \le 64$            then $curvature = C / 100000$
if $64 < |C| \le 128$      then $curvature = 2 * (C - \text{SIGN}(C) * 32) / 100000$
if $128 < |C| \le 192$     then $curvature = 4 * (C - \text{SIGN}(C) * 80) / 100000$
if $192 < |C| \le 256$     then $curvature = 8 * (C - \text{SIGN}(C) * 136) / 100000$
if $256 < |C| \le 320$     then $curvature = 16 * (C - \text{SIGN}(C) * 196) / 100000$
```

if $320 < |C| \le 384$ then *curvature* = $32 * (C - \text{SIGN}(C) * 258) / 100000$
if $384 < |C| \le 448$ then *curvature* = $64 * (C - \text{SIGN}(C) * 321) / 100000$
if $448 < |C| \le 511$ then *curvature* = $128 * (C - \text{SIGN}(C) * 384) / 100000$

9.2 Route Number Types

Table 31: Route numbers types for selected countries

Country	1	2	3	4	5	6
Andorra	European E#	Autopista A#	Carretera General CG#	Carretera Secundràia CS#		
Australia	National Highway (1) NI#,I#	National Route (2) N#	State Route (3) N# State Highway (5) S#	Metro Road (4) X#,M#,MI#,F#	Metro Road RIC/MGM (6) MI# AI Road (8) AI#	M-Road (7) M# A Road (9) A# B Road (10) B# C Road C# Tourist Roads are coded as named

Country	1	2	3	4	5	6
Austria	European E#	Autobahnen A#	Schnellstraßen S#	Bundesstraßen B#	Landesstraßen L# Landeshauptstraßen LH#	#
Bahrain	no route numbers exist					
Belgium	European E#	Autosnelwegen A#	Gewestwegen N#			B# R#
Brazil	Federal BR-#	State SP-#				
Canada	Primary Provincial Routes HWY-# Autoroutes AUT-#	Secondary Provincial Routes HWY-# Autoroutes RTE-#	Regional RR-# County CR-# District DR-#			
Czech Republic	European E#	Dálnice D#	Silnice I. tridy #	Silnice II. tridy #	Silnice III. Tridy #	
Denmark	European E#	Primærrute #	Sekundærrute #			
France	European E#	Autoroutes A#	Routes Nationales N#	Routes Départementales D#	Routes Communales C# Routes Vicinales V# Voies Rurales R#	B# M# T#

Country	1	2	3	4	5	6
Finland	European E#	Valtatie #	Kantatie #	Seututie #		
Germany	European E#	Autobahn A#	Bundesstraßen B#	Landesstraßen L# Staatsstraßen S# ST#	Kreisstraßen K#	F#
Ireland	Motorway M#	National N#	Regional R#			
Italy	European E#	Autostrada A#	Strada Statale SS#	Strada Provinciale SP#		
Kuwait	numbers only					
Liechtenstein	European E#	Autostraßen A#	Hauptstraßen #			
Luxembourg	European E#	Autoroutes A#	Routes National N#	Chemins Repris CR#		
Monaco	European E#	Autoroutes A#	Routes Nationales N#	Routes Départementales D#	Routes Communales C# Routes Vicinales V# Voies Rurales R#	B# M# T#
Netherlands	European E#	Autosnelwegen A#	Nationale Wegen N#			R# S#
Norway	European E#	Riksveg #	Fylkeveg #			

Country	1	2	3	4	5	6
Oman	no route numbers exist					
Portugal	European E#	Autoestrada A#	Itenerário Principal IP#	Itenerário Complementar IC#	Estrada Nacional N#	Estrada Municipal M#
San Marino	European E#	Autostrada A#	Strada Statale SS#	Strada Provinciale SP#		
Saudi Arabia	2 digit number #	3 digit number #	4 digit number #			
Singapore	no route numbers exist					
Slovak Republic	European E#	Dialnice D#	Cesty I. triedy #	Cesty II. triedy #	Cesty III. triedy #	
South Africa	National N#	Regional R#	Metro M#			
Spain	European E#	Autopista A#	Carretera Nacional N#	Carretera Comarcal C#	Carretera Local Provincial 1 st letter of Province + P# Carretera Local Vecinal 1 st or 2 nd letters of Province + V#	
Sweden	European E#	Riksväg #10-99	Länsväg #100-499			
Switzerland	European E#	Autostraßen A#	Hauptstraßen #			
Taiwan	Federal (1 digit) #	Provincial (2 digits) #	3 digits #			
UK	European E#	Motorways M#	A-Roads A#	B-Roads B#		

Country	1	2	3	4	5	6
United Arab Emirates	E E#	Dubai D#	Sharjah S#			
US	Interstate I-#	Federal US-#	State	County CR-#		
Vatican City	European E#	Autostrada A#	Strada Statale SS#	Strada Provinciale SP#		

9.3 Heading Change Definition

Heading Changes are changes in direction along a path. If the road geometry is represented as poly-line, Heading Changes can be calculated by taking 3 consecutive shape points and calculating the line segment angles between the first and the second shape point and the second and third respectively. The angle between the first and the second line segment is called Heading Change.

Following figures illustrate Heading Changes:

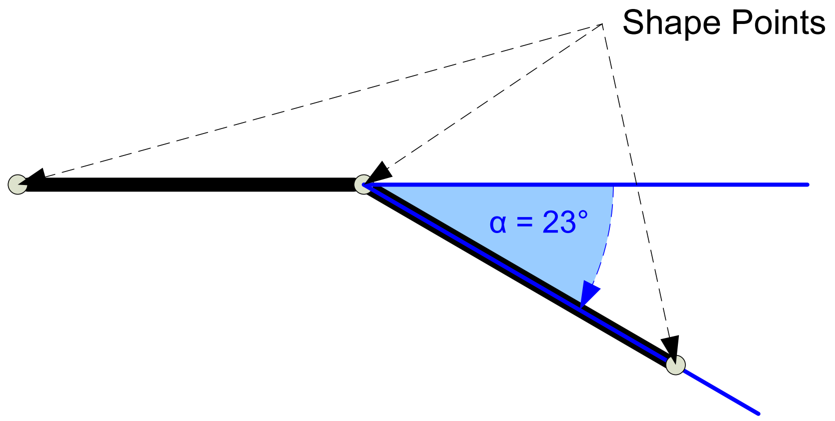


Figure 39: Calculation of Heading Change angle

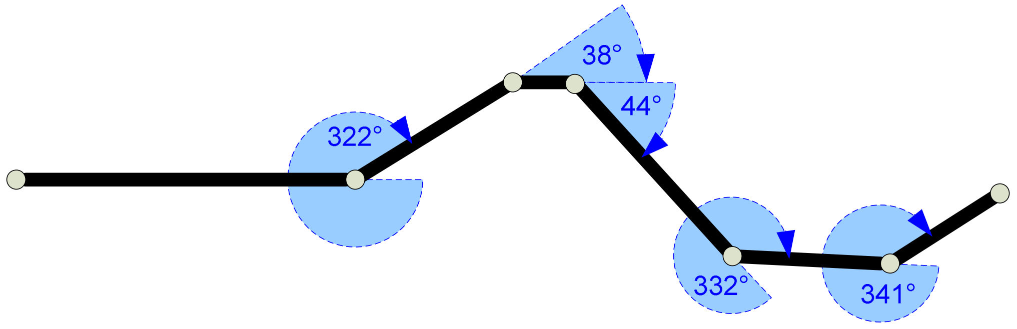


Figure 40: Heading Changes along a Path

9.4 Coding Scheme for “Region Code”

The value in the field “Region Code” is based on ISO 3166-2:2007 "*Codes for the representation of names of countries and their subdivisions - Part 2: Country subdivision code"*. This standard defines codes for all relevant administrative subdivisions of all countries. The country subdivision code consists of two parts “CC-SSS”: The first part “CC” is the alpha-2 code element from ISO 3166-1 representing the country and is already covered in the META-DATA message by field “Country Code”, the second part “SSS” represents the top-level subdivision of this country.

The subdivision code part “SSS” uses up to three alphanumeric characters. To represent this code a field of length 24 bits assuming 8 bits per character is needed. All subdivision codes defined by ISO 3166-2 use either digit characters (“0”…”9”) or uppercase alphabetic characters (“A”…”Z”). To minimize the needed field length, each subdivision code character can be represented by the codes 1…26, a missing character, a space character or the digit “0” can be represented by code 0. This code range can be represented by 5 bits according to Table 32 assuming that the subdivision code part of ISO 3166-2 is either completely numeric or completely alphabetic to make the resulting code unique. A review of the ISO 3166-2 document showed, that currently all countries uses either a completely numeric or a completely alphabetic scheme for the top-level regions.

The value of the field “Region Code” is encoded using the following mapping scheme:

Step 1: The upto three characters of the subdivision code part of ISO 3166-2 are first aligned and padded to a code of three characters: If all ISO 3166-2 code characters are digits, they are right-aligned and padded with “0” on the left. In all other case (i.e. the ISO 3166-2 code characters contain at least one alphabetic character), they are left-aligned and filled with space characters on the right. (The different alignment helps to avoid code clashes if numeric and alphabetic subdivision codes exist for the same country.)

Step 2: The resulting three characters are coded according to Table 32. The three 5 bit codes are packed in the region code field of 15 bits length by the formula $((code_{char1} * 32 + code_{char2}) * 32 + code_{char3})$, i.e., by shifting the code of the first character by 10, the code of the second character by 5, the code of the third character by 0 bits to the left.

The value of the field “Region Code” is decoded accordingly. Because the encoding of the alphabetic characters “A”…”J” and the numeric characters “1”…”9” uses the same 5 bit codes, the decoded subdivision code can be ambiguous in very rare cases.

Encoding Examples:

ISO 3166-2 subdivision code of “Hessen” in “Germany”: “DE-HS” – “DE” is represented by a country code of 276 (ISO 3166-1 numeric) and the padded subdivision code “HS_” is encoded by the region code of $(((8_{\text{for “H”}} * 32 + 20_{\text{for “S”}}) * 32 + 0_{\text{for Space}}) = 8832$.

ISO 3166-2 subdivision code of “Vorarlberg” in “Austria”: “AT-8” – “AT” is represented by a country code of 40 and “008” is encoded by the region code of $(((0_{\text{for “0”}} * 32 + 0_{\text{for “0”}}) * 32 + 8_{\text{for “8”}}) = 8$.

ISO 3166-2 subdivision code of “Alsace” in “France”: “FR-A” – “FR” is represented by a country code of 250 and “A__” is encoded by the region code of $(((1_{\text{for “A”}} * 32 + 0_{\text{for Spaces}}) * 32 + 0_{\text{for Space}}) = 8192$.

ISO 3166-2 subdivision code of département “Ain” in “France”: “FR-01” – “FR” is represented by a country code of 250 and “001” is encoded by the region code of $(((0_{\text{for “0”}} * 32 + 0_{\text{for “0”}}) * 32 + 1_{\text{for “1”}}) = 1$.

ISO 3166-2 alphabetic character	***ISO 3166-2 numeric character***	***5 bit code in field “Region Code”***
none/Space	***none/Space/0***	0
A	***1***	1
B	***2***	2
C	***3***	3
D	***4***	4
E	***5***	5
F	***6***	6
G	***7***	7

ISO 3166-2 alphabetic character	***ISO 3166-2 numeric character***	***5 bit code in field "Region Code"***
H	***8***	8
I	***9***	9
J		10
K … Y		11 … 25
Z		26
		27…31 – not used

Table 32: Coding Scheme for field "Region Code" to map each ISO 3166-2 alphanumeric character to a 5 bit code

10 References

[1] http://www.ertico.com/en/subprojects/adasis_forum/objectives_approach/
[2] http://www.prevent-ip.org/en/prevent_subprojects/horizontal_activities/maps__adas/
[3] Mezger K, et al (2004), "Preliminary Interface Requirements, Functional Architecture and Logical Data Model" D12.32, MAPS & ADAS Consortium and ADASIS Forum, Brussels
[4] Otto H.U, et al (2005), "Data Requirements" D1, MAPS & ADAS Consortium and ADASIS Forum, Brussels
[5] Angenvoort J, et al (2005), "Draft Specifications", Deliverable D3, MAPS & ADAS Consortium and ADASIS Forum, Brussels
[6] Angenvoort J, et al (2005), "Interface and Data Entity Specifications (Draft): Part 1 – General Specifications", Deliverable D12.42.1, MAPS & ADAS Consortium and ADASIS Forum, Brussels
[7] Beuk L, et al (2005), "Interface and Data Entity Specifications (Draft): Part 2 – Data Protocol", Deliverable D12.42.1, MAPS & ADAS Consortium and ADASIS Forum, Brussels
[8] NAVTEQ's GDF 3.0 Reference Manual v20.0, NAVTEQ 2006
[9] Tele Atlas Multinet Version 3.5 Format Specification, Data Model, and Data Specification, Tele Atlas NV, 2008.
[10] CEN ISO 14825:1995, "Geographic Data Files (GDF) 3.0", 1996.
[11] ISO 14825:2004, "Geographic Data Files (GDF) 4.0", 2004.
[12] ISO 14825:20xx, "Geographic Data Files (GDF) 5.0", Draft, March 2009.
[13] NDS Version 2.1.1.1 Navigation Data Standard Specification, April 2010.
[14] Durekovic S, et al (2009), "ADASRP 2009 User's Manual", NAVTEQ 2009.
[15] ISO 3166-1:2006, "Codes for the representation of names of countries and their subdivisions - Part 1: Country code"
[16] ISO 3166-2:2007, "Codes for the representation of names of countries and their subdivisions - Part 2: Country subdivision code"