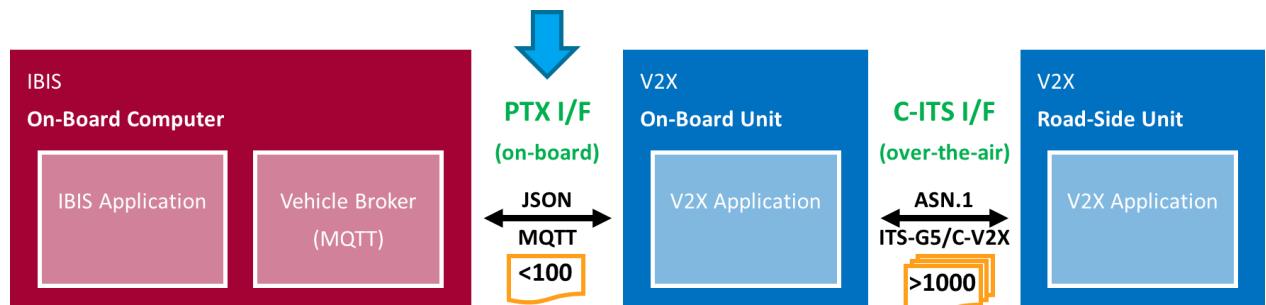


C-ITS for Public Transport Vehicles

PTX - Interface between IBIS and OBU



Specifications

Version 2B

This document is edited by:

© 2025 Trapeze Switzerland GmbH. All rights reserved.

Trapeze Switzerland GmbH
Rheinstrasse 36 | 8212 Neuhausen | Switzerland
phone +41 58 911 11 11 | fax +41 58 911 11 12
Info.ch@trapezegroup.com | www.trapezegroup.com

Table of Contents

Table of Contents	2
Table of Figures.....	5
Document Management.....	6
1 Introduction.....	7
1.1 Purpose	7
1.2 Acronyms.....	7
1.3 Informative References	9
2 From Current Traffic Signal Priority to V2X-based Signal Phase Assistance.....	10
2.1 Initial Situation	10
2.2 Restructuring of Analog Frequencies in Germany	10
2.3 Migration from Analog Radio to C-ITS	11
2.3.1 Stage 1A: Transitory Solution.....	11
2.3.2 Stage 1B: Transitory Solution “Plus”	11
2.3.3 Stage 2A: Traffic Light Priority by means of V2X.....	12
2.3.4 Stage 2B: Signal Phase Assistance by means of V2X	12
2.4 Future Extensions.....	12
2.4.1 Dynamic Speed Limits	13
2.4.2 Warnings about Emergency Vehicles	13
2.4.3 Constriction Zone Coordination	13
2.4.4 Safety of Vulnerable Road Users	13
2.5 Summary	13
3 Architecture Overview	14
3.1 Objectives	14
3.2 Vehicle Compositions	14
3.2.1 Single Uni-Directional Vehicle	14
3.2.2 Coupled Uni-Directional Vehicles.....	15
3.2.3 Single Bi-Directional Vehicle	15
3.2.4 Coupled Bi-Directional Vehicles	16
3.3 Security Considerations	17
4 Use Cases and Enabling V2X Messages	18
5 Protocol.....	19
5.1 Message Transport.....	19
5.2 Message Encoding	19
5.3 Message Header	20
5.4 Message Topics	21
5.4.1 Topic Prefixes	21
5.4.2 Topic Directory.....	21
5.5 Message Expiration	22

5.6 Message Repetition.....	22
5.7 Message Attributes.....	22
6 Device Management and Logging.....	23
6.1 Enumeration Types	23
6.1.1 Module Class	23
6.1.2 Reachability	23
6.1.3 Activation	23
6.1.4 Health.....	24
6.1.5 Trigger.....	24
6.1.6 Log Level	24
6.1.7 Power State	25
6.2 IBIS-to-Device Messages.....	26
6.2.1 Power State	26
6.2.2 Log Level Configuration.....	27
6.2.3 Command Trigger.....	27
6.3 Device-to-IBIS Messages.....	28
6.3.1 Power Request	28
6.3.2 Log Messages	29
6.3.3 Device Version Info	30
6.3.4 Device Health Info	31
7 Operational Information.....	33
7.1 Enumeration Types	33
7.1.1 Vehicle Category	33
7.1.2 Location Status	33
7.1.3 Priority Level	33
7.1.4 Driver Cab Activation	34
7.2 IBIS-to-Device Messages.....	35
7.2.1 Vehicle Info	35
7.2.2 Operational Logon	37
7.2.3 Operational Journey	38
7.2.4 Operational Status	40
8 V2X-specific Messages.....	44
8.1 Enumeration Types	44
8.1.1 Encoding Rule	44
8.1.2 Service Type.....	44
8.1.3 Message Type	45
8.1.4 Movement Phase State	45
8.1.5 Priority Status	46
8.2 IBIS-to-OBU Messages.....	47
8.2.1 Configuration	47
8.2.2 Path Definition	49
8.2.3 Path Location.....	51
8.2.4 R09 Request.....	52

8.3 OBU-to-IBIS Messages	53
8.3.1 Capabilities	53
8.3.2 R09 Response	55
8.3.3 Intersection Map	56
8.3.4 Intersection Phase	60
8.3.5 Intersection Status	63
8.4 Air Interface Mirroring	65
8.4.1 Sent Messages	66
8.4.2 Received Messages	66
8.4.3 PCAP Dumps	66
9 Interaction Diagrams.....	67
9.1 Sending CAM [stage 0]	67
9.2 Sending R09 over CAM [stage 1A]	68
9.3 Sending R09 over SRM [stage 1B]	68
9.4 Sending SRM with ETA [stage 2A]	69
9.5 Signal Phase Assistance [stage 2B]	71
Appendix A – Protocol Buffer Schema Definitions	72
A.1 Framing (PTX_Framing_DTO.proto)	72
A.2 Device Management (PTX_DeviceManagement_DTO.proto)	72
A.3 Operational Information (PTX_OperationalInfo_DTO.proto)	74
A.4 Vehicle to Everything (PTX_VehicleToEverything_DTO.proto)	75
Appendix B – Schema Files	79
Appendix C – Illustrations	80
C.1 Path Definition.....	80
Appendix D – Sources for Interaction Diagrams	81
D.1 Sending CAM	81
D.2 Sending R09 over CAM	81
D.3 Sending R09 over SRM	81
D.4 Sending SRM with ETA	82

Table of Figures

Figure 1: Traffic Signal Control According to VDV420.....	10
Figure 2: Conventional Traffic Signal Control via Analog Radio.....	10
Figure 3: Traffic Signal Control via V2X (transitory solution).....	11
Figure 4: Traffic Signal Control via V2X (transitory solution "plus").....	11
Figure 5: Traffic Signal Control via V2X	12
Figure 6: Traffic Signal Control and Traffic Signal Phase Assistance via V2X.....	12
Figure 7: Architecture Overview.....	14
Figure 8: Single Uni-Directional Vehicle	15
Figure 9: Coupled Uni-Directional Vehicles	15
Figure 10: Single Bi-Directional Vehicle.....	16
Figure 11: Coupled Bi-Directional Vehicle	16
Figure 12: Interaction Diagram "Sending CAM".....	67
Figure 13: Interaction Diagram "Sending R09 over CAM"	68
Figure 14: Interaction Diagram "Sending R09 over SRM"	69
Figure 15: Interaction Diagram "Sending SRM with ETA"	70
Figure 16: Complex Path Setup	80

Document Management

Contact Persons	
Herman	Ivo Herman – ivo.herman@herman.cz
IVU	Ingmar Gebhardt – ige@ivu.de
Trapeze	Dominique Müller – dominique.mueller@trapezegroup.com (editor)
Yunex	Fabian Riether – fabian.riether@yunextraffic.com

Version History				
Version	Date	Name	Description of Changes	
1-1W	27.08.2024	mdo	Dominique Müller	draft submission to VDV435 group
1X	17.09.2024	mdo	Dominique Müller	after adding IVU to specs team
1Y	27.09.2024	mdo	Dominique Müller	minor changes agreed in last meeting
1Z	11.02.2025	mdo	Dominique Müller	changed type of PathID int64 → string changed definition of Dist in PathLocation message
2	15.02.2025	mdo	Dominique Müller	made bools optional in IntersectionMap message PathID → PathLocation in IntersectionStatus message added appendix with illustrations
2A	21.02.2025	mdo	Dominique Müller	added description of topic prefixes and topic directory removed ambiguity regarding “version” in header
2B	22.02.2025	mdo	Dominique Müller	added bridge (to backend) to the topic directory

1 Introduction

1.1 Purpose

The BAST proposes in its recommendation "Nutzung der C2X-basierten ÖV-Priorisierung an signalisierten Knotenpunkten" [1] a stepwise implementation of C-ITS (V2X).

In its COSEL project [2], the DVB (Dresdener Verkehrsbetriebe) and the TU-Dresden (Technical University of Dresden) have implemented a signal phase assistant that helps the drivers to optimise their driving behaviour, leading to significant energy savings and increasing passenger satisfaction thanks to optimising the calls at stops.

The purpose of this document is to specify the message exchanges between an IBIS (on-board unit for AVL operation) and a V2X OBU (on-board unit communicating with the RSU), where the scope for the first version of this document is derived from the BAST recommendation and the COSEL project.

1.2 Acronyms

Term or Acronym	Definition
BAST	Bundesanstalt für Straßenwesen (= Federal Roads Office) in Germany
C-ITS	Cooperative Intelligent Transportation System: Synonym for V2X
C-V2X	Cellular V2X: 3GPP standard for V2X communication based on cellular technology
CAM	Cooperative Awareness Message: V2X message on the vehicle type, location, direction of travel and speed with which the vehicle draws attention to its presence and intention (can transport an R09.16 telegram as a transitory solution)
COSEL	Computer-Optimised Speed control for Energy-efficient Light-rails: The name of a project to save energy and optimise the timing at stops thanks to driver assistance based on real-time data of traffic signal control. Also, a countess at the Dresden court in the 18 th century.
ETA	Expected time of arrival (of the vehicle at the stop line of the junction)
ETSI	European Telecommunication Standards Institute: The body standardising C-ITS (V2X) communication in Europe. Builds on J2735.
IBIS	Integrated On-Board Information System: On-board computer installed in the vehicle by the ITCS supplier. From German: Integriertes Bord-Informations-System
ISO	International Standardisation Organisation: The body defining and maintaining global standards
ITS-G5	Intelligent Transport Systems G5: ETSI standard for V2X communication based on IEEE 802.11p, therefore also known as "WIFI-p".
J2735	SAE message dictionary for surface vehicle communication

Term or Acronym	Definition
JSON	JavaScript Object Notation
MAP	Map Message: V2X message that informs approaching vehicles of the layout of the junction with its lanes, connections, and signal groups
MQTT	Message Queue Telemetry Transport: Communication protocol used in the IoT space
OBU	<p>On-board unit: V2X device in the vehicle that communicates with the RSU on the track and is connected to the on-board computer</p> <p>Note: In the ETSI specifications relevant for V2X (C-ITS), the terms OBU and RSU are consistently used for such devices; this terminology is upheld in this document to avoid confusion</p>
PCAP	Packet Capture: file format for network packet analysis; often used with the Wireshark analysis tool (of which later versions support J2735)
R09.16	Record 09 (16-byte version): A telegram originally defined in VDV specification 420 (a large number of variants exist), which is used for conventional traffic signal control via analogue or digital radio
RSU	<p>Road-side unit: V2X device at the junction that communicates with the OBU in the vehicle and is connected to the junction controller (though in general not limited to junction controllers)</p> <p>Note: In the ETSI specifications relevant for V2X (C-ITS), the terms OBU and RSU are consistently used for such devices; this terminology is upheld in this document to avoid confusion</p>
SAE	Society of Automotive Engineers: The body standardising C-ITS (V2X) communication in the United States
SPAT	Signal Phase and Timing Message: V2X message (related to MAP), which informs approaching vehicles of the current phase and the forecasts for the next phases for each signal group at an intersection
SRM	Signal Request Message: V2X message (related to MAP) with which the vehicle requests its prioritisation (can transport an R09.16 telegram as a transitory solution)
SSM	Signal Status Message: V2X message (related to SRM) with which the traffic computer communicates the status of prioritisation
V2X	Vehicle-to-Everything Communication: new type of digital communication based on either "C-V2X" or "ITS-G5"
XML	Extensible Markup Language

1.3 Informative References

# (DM#)	Document
1	BAST - Nutzung der C2X-basierten ÖV-Priorisierung an signalisierten Knotenpunkten https://bast.opus.hbz-nrw.de/files/2595/V353+BF+Gesamtversion.pdf
2	COSEL - Computer-Optimised Speed control for Energy-efficient Light-rails https://its-mobility.de/wp-content/uploads/III_05_Gassel_DVB_C-ITS-Forum_2024.pdf

2 From Current Traffic Signal Priority to V2X-based Signal Phase Assistance

This chapter is INFORMATIVE.

2.1 Initial Situation

As per the year 2024, conventional traffic signal priority via analogue or digital radio based on VDV regulations 420 and 426 is based on pre-supplied reporting point chains: When passing a reporting point, the vehicle transmits an R09.16 telegram, which is transferred from the road-side radio receiver to the traffic signal controller.

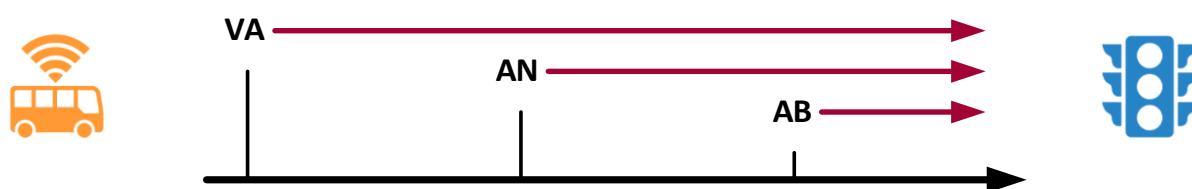


Figure 1: Traffic Signal Control According to VDV420

These telegrams represent either a pre-announcement (VA), an announcement (AN) or a cancellation (AB). Based on the information contained in the telegram, including the timetable deviation and the line, the traffic signal controller determines which priority the vehicle is given, and which lane is to be activated.

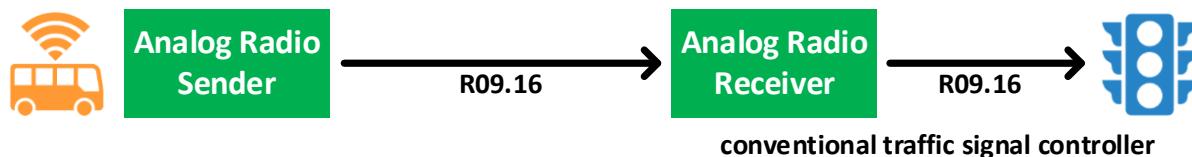


Figure 2: Conventional Traffic Signal Control via Analog Radio

The preparation of reporting point chains is time-consuming and has the disadvantage that the control centre cannot flexibly plan the routes while prioritising the vehicles at the same time, as the reporting point chains always activate the planned lane, even if another lane would be the correct one due to re-routing. Due to the lack of feedback, the driving personnel remain in the dark as to whether the prioritisation has been successful.

2.2 Restructuring of Analog Frequencies in Germany

At the end of 2028, the analogue radio frequencies used for traffic signal priority in Germany will be restructured: The channel grid will be reduced from 20 kHz to 12.5 kHz and some of the allocated frequencies will be moved to other frequency bands.

As a result, the analogue radio equipment (transmitters in the vehicles, receivers at the roadside) will have to be reprogrammed or even replaced. Various local authorities have decided to no longer invest in this technology, which dates back to the 1980s. Instead, they are planning to switch to modern "vehicle-to-everything" technology (V2X).

V2X is not limited to public transport and vehicles belonging to emergency services. In fact, private vehicles are already being equipped with it to communicate with each other and with the infrastructure. The digitalisation of traffic signal control is thus becoming a springboard for other applications that V2X technology makes possible.

2.3 Migration from Analog Radio to C-ITS

This technological quantum leap offers many opportunities but requires careful planning. The old and new technology must be kept in operation in parallel for several years.

In its recommendation for action "Use of C2X-based public transport prioritisation at signalised junctions ([1] chapter 3.3)", the Federal Roads Office (BAST) recommends the changeover in two stages:

- Stage 1: Prioritisation using "R09.16 embedded in CAM"
- Stage 2: Prioritisation using SRM/SSM (= "pure V2X")

Stage 1 is an interim solution that allows the V2X radio link (OBU in the vehicle, RSU at the intersection) to be rolled out quickly without having to replace the intersection controllers. The rollout can take place before the restructuring of the analogue radio frequencies takes effect.

Stage 2 is the desired state. The intersection controllers have been upgraded and are therefore ready for V2X. The full potential can be realised. The traffic signal phase assistant and other intelligent functions can now be implemented.

2.3.1 Stage 1A: Transitory Solution

As recommended by the BAST, the radio link is replaced by the tandem OBU ↔ RSU in the transitory solution.

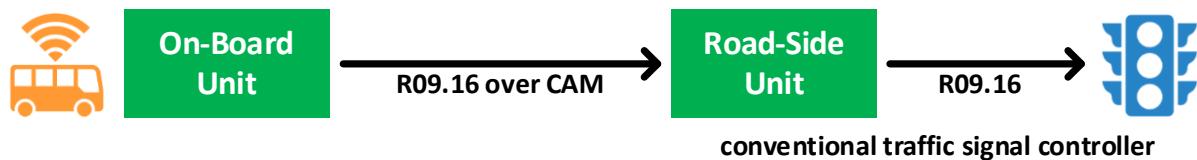


Figure 3: Traffic Signal Control via V2X (transitory solution)

Reporting point chains still must be supplied and the driver does not receive any feedback on prioritisation.

2.3.2 Stage 1B: Transitory Solution “Plus”

As part of the C-ROADS projects in Germany, the embedding of R09.16 telegrams in an SRM was also tested as an alternative to CAM. SRM can be relayed among V2X units (RSUs and OBUs) using “geo routing”, extending the reach. In addition, the SSM is available as a response.

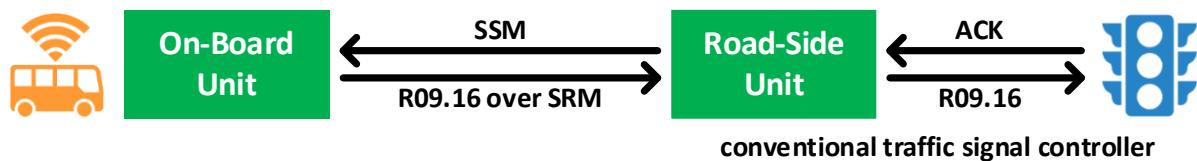


Figure 4: Traffic Signal Control via V2X (transitory solution “plus”)

This extension of stage 1A to include a response is also based on the R09.16 telegram. This means that the signalling point chains supplied are still required.

Thanks to the feedback, it is possible to display on the vehicle control unit whether the prioritisation request has been received by the intersection computer and whether it has been accepted or rejected.

2.3.3 Stage 2A: Traffic Light Priority by means of V2X

As with stage 1B, in stage 2 the request is made using SRM, but without embedding an R09.16 telegram. This is made possible by a V2X-compatible traffic signal controller that transmits the MAP (= layout of the junction and its lanes). The SRM refers to these lanes and therefore no longer requires any reporting point chains.

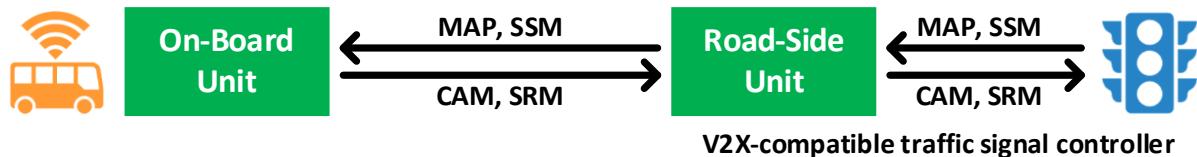


Figure 5: Traffic Signal Control via V2X

"The changeover from the familiar reporting point principle to quasi-continuous tracking of a public transport vehicle via V2X and sending SREMs to and from a traffic light node requires a rethink by traffic engineers and the expansion of existing control procedures or the development of new ones," writes the BAST in chapter 3.3 of its recommendation [1].

The transport operator can now save itself the time-consuming and tedious supply of the reporting point chains and gains flexibility in scheduling, as the public transport lines are no longer statically stored in the junction computer.

From the traffic engineers' point of view, the opportunity lies in the fact that valuable green phases are no longer wasted because the vehicles are permanently tracked.

2.3.4 Stage 2B: Signal Phase Assistance by means of V2X

Not only the SRM is based on the MAP, but also the SPAT. The OBU obtains information on when the signal changes from red to green or how long the green phase will last.

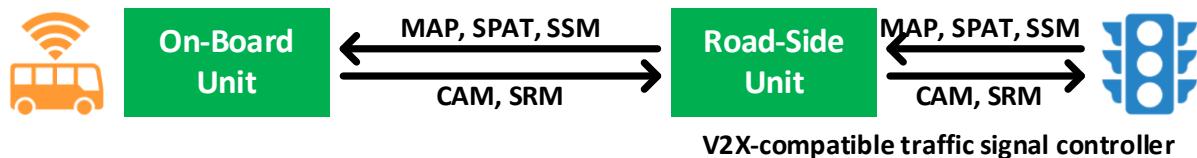


Figure 6: Traffic Signal Control and Traffic Signal Phase Assistance via V2X

The OBU forwards this information to the on-board computer, which uses it to derive driving recommendations for the driver and displays them on the driver terminal.

This allows the following two use cases to be realised – to the benefit of the transport operator:

1. energy-optimised approach to the junction
2. time-optimised departure from the stop

2.4 Future Extensions

V2X will over time evolve to enable more and more use cases, to increase security and smoothen the traffic flow. Without claiming completeness, a few of these use cases are sketched herein. To support such additional use cases, this specification will need to be extended.

2.4.1 Dynamic Speed Limits

Drivers of public transport vehicles can be informed about dynamic speed limits. The IBIS can use this information in various ways, e.g. for ETA calculations at the next stop, for routing decisions, or for speeding alerts to the driver.

2.4.2 Warnings about Emergency Vehicles

Drivers of public transport vehicles can be warned when emergency vehicles (police, fire fighters, ambulance) are approaching. This is not limited to intersections.

2.4.3 Constriction Zone Coordination

Buses often must coordinate when approaching a narrow passage or any other kind of constriction zone. This functionality is today available to coordinate between public transit buses, but private trucks may still cause buses getting stuck. Thanks to V2X, also the private vehicles can participate in constriction zone coordination.

2.4.4 Safety of Vulnerable Road Users

Intersections equipped with cameras or LIDARs can identify vulnerable road users (pedestrians, cyclists, bikers) and relay this information to vehicles. Drivers of public transport can thus be warned if VRUs are nearby.

2.5 Summary

V2X requires cooperation between the local authority (for converting the traffic light systems) and the transport operator (for converting the vehicles). The effort is well worth the while: Once fielded, many other interesting applications can be realised based on V2X technology.

3 Architecture Overview

The IBIS on-board computer hosts an MQTT broker. This broker is the communication hub for all on-board communication in the future. This document is foreseen to be contributed into the VDV-435 standard (aka. IoM, Internet of Mobility).

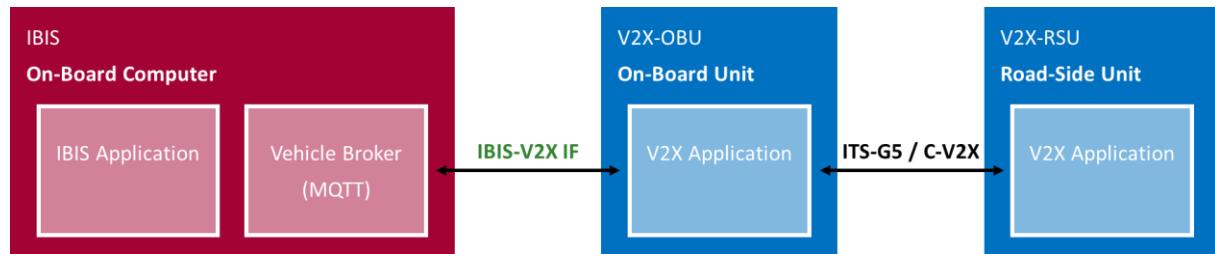


Figure 7: Architecture Overview

The V2X on-board unit (OBU) brokers between the “V2X world” and the “VDV world”. The V2X specifications have been conceived with a plethora of road users and use cases in mind. The services and information elements that are relevant to public transport are visible on the VDV side of the OBU.

3.1 Objectives

This “IBIS-V2X” interface specification has been created with the following objectives in mind:

- the IBIS is shielded from the complexities of the V2X (C-ITS) specs
- evolution of the V2X spec should not impact the VDV specs

Further requirements that the V2X module is expected to fulfil:

- support ETSI ITS-G5 (aka. IEEE 802.11p) or 3GPP C-V2X or both
- support firmware upgrade without on-board intervention
- follow the security best practices as per chapter 3.3

3.2 Vehicle Compositions

This section illustrates various deployment scenarios. The purpose of these scenarios is to show how V2X OBUs can be deployed to uni-directional and bi-directional vehicles (the latter might be fitted with two OBUs) and that the vehicles may or may not be coupled to form train compositions.

3.2.1 Single Uni-Directional Vehicle

This situation is a typical bus deployment. It may also arise when a uni-directional rail car is currently not coupled with any other cars.

«forward» direction

Figure 8: Single Uni-Directional Vehicle

There is only a single driver cab “A”, which may be occupied or unoccupied. The “forward driving direction” is always the same. Positive speed denotes forward motion, negative speed denotes backward motion. The geo location of the vehicle indicates the front of the driver cabin.

3.2.2 Coupled Uni-Directional Vehicles

This situation arises when uni-directional rail cars are coupled to form a train composition.

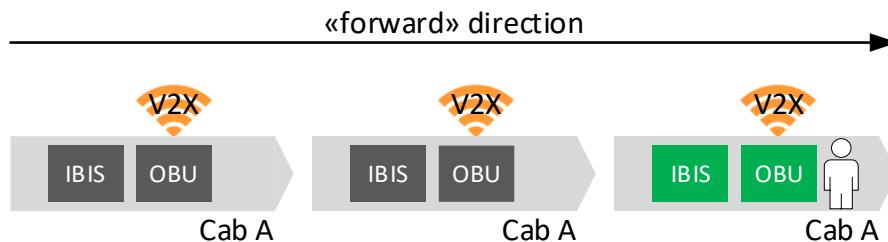


Figure 9: Coupled Uni-Directional Vehicles

Each car has a driver cab “A”, but only the front one may be occupied or unoccupied. The “forward driving direction” is always the same. Positive speed denotes forward motion, negative speed denotes backward motion. The geo location of the vehicle indicates the front of the active driver cabin.

On all vehicles, the IBIS is expected to be wired to the “driver cab” signal. It shall provide this information to the OBU by means of the “OperationalStatus” message (c.f. chapter 7.2.4), in order to avoid extra cabling. In some cases, the OBU may be wired to the “driver cab” signal directly.

Within the train, only a single OBUs may be “active” at any given time.

- active: the OBU may exchange V2X messages (in particular CAM and SRM) with the RSU and MQTT messages pertaining to V2X (c.f. chapter 8) with the IBIS
- inactive: the OBU may only receive V2X messages from the RSU and MQTT messages from the IBIS, with the exception of messages pertaining to device management and logging (c.f. chapter 6) which it shall still send as specified

3.2.3 Single Bi-Directional Vehicle

This situation arises when a bi-directional rail car is currently not coupled with any other cars.

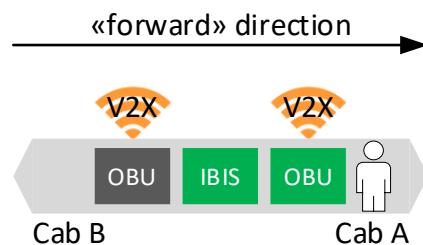


Figure 10: Single Bi-Directional Vehicle

The vehicle has both a driver cab "A" and a driver cab "B", but only one of them may be occupied or unoccupied. The "forward driving direction" changes as a function of the occupied driver cab. Forward (positive speed) and backward (negative speed) motion are expressed with respect to the occupied driver cab. The geo location of the vehicle indicates the front of the active driver cabin.

There may or may not be two OBUs in a bi-directional vehicle. If there are two OBUs, then they will be assigned to one driver cab each. Within the vehicle, only one of the two OBUs may be active at any given moment.

The IBIS is expected to be wired to the "driver cab" signals. It shall provide this information to the OBU, in order to avoid extra cabling. In some cases, the OBUs may be wired to the "driver cab" signals directly.

3.2.4 Coupled Bi-Directional Vehicles

This situation arises when bi-directional rail cars are coupled to form a train composition.

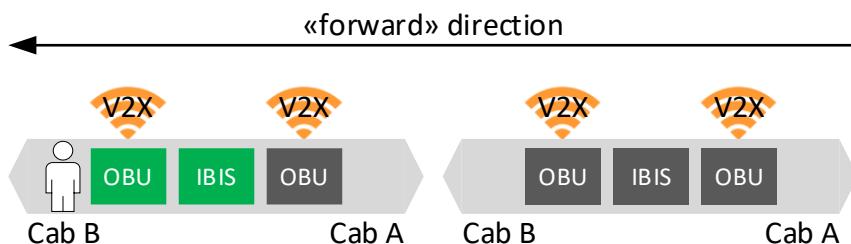


Figure 11: Coupled Bi-Directional Vehicle

No matter how many vehicles, and how many driver cabins per vehicle, only one of the driver cabins, on either end of the train, will be occupied. As in the other depicted scenarios, the IBIS is expected to be wired to the "driver cab" signals and to provide this information to the OBU.

The figure shows a change in the "forward direction". This means that geo location and speed of the vehicle are now expressed in terms of the active driver cab "B".

Note: Upon a head change of a train, the geo location can teleport dozens or even hundreds of metres.

3.3 Security Considerations

Both the IBIS and the OBU require protection:

- protecting the IBIS protects by extension the operations control backend, which is considered a critical infrastructure
- protecting the OBU protects by extension the air interface: the aim is to minimise the risk of an attacker taking over the OBU and manipulating traffic control or even causing safety-relevant situations

The following best practices¹ should be implemented (applies to MQTT broker, MQTT client libraries, and application software of both the IBIS and the OBU):

- implement secure authentication and authorization
- enable encryption
- regularly update and patch
- monitor for anomalies
- follow secure coding practices

As far as this specification is concerned, the point of control from security perspective is the MQTT broker. Wherever it resides, on the IBIS, on the OBU, or elsewhere, the clients (the IBIS application, the OBU application) need to establish a secure connection (TLS) and to authenticate themselves (username+password or client certificate).

Furthermore, it must be ensured that:

1. only authenticated devices can connect to the broker
2. the connected devices communicate only as per the respective specifications (in this case PTX), and in particular can not
 - o subscribe to any messages (on topics) that are none of their business
 - o publish any messages (to topics) that they are not entitled to publish
3. the devices validate the received messages before processing

How exactly this is achieved will depend on the MQTT broker product and on the MQTT client library. Typically, there will be a “user account” on the broker for each authorised device. This user account has associated read/write permissions on parts of the topic tree.

As a first guard against buffer overflow attacks (which could open the door to injection of malicious code), on the broker, there will be a configured maximum message size. The clients shall protect themselves to be immune to oversized messages or to messages that do not fit the defined schema.

Whether username+password or client certificates are used for authentication, how often the credentials are refreshed, or how the process of updating them works, will depend on the project or customer.

¹ https://medium.com/@_crac/understanding-mqtt-cves-vulnerabilities-and-best-practices-b9f71c44f6da

4 Use Cases and Enabling V2X Messages

This chapter is INFORMATIVE.

This chapter identifies the use cases (and their enabling V2X messages) that are addressed in the current scope of this specification.

Use Case	V2X Messages
Stage 1A Traffic signal priority transitory solution	R09.16 over CAM As intended by BAST, the R09.16 telegram according to VDV 420 (or proprietary variants thereof) can be embedded in a CAM message and sent to the RSU, thereby replacing the analogue radio connection with C-ITS communication.
Stage 1B Traffic signal priority transitory solution	R09.16 over SRM with SSM response The R09.16 telegram can be embedded in an SRM and be acknowledged by an SSM.
Stage 2A Traffic signal priority for public transport vehicles	MAP, SRM, SSM Public transport vehicles request priority by means of the SRM-SSM handshake. If SSM is not received or if the expected time of arrival (ETA) at the intersection changes, the SRM is resent. The SRM references the MAP.
Stage 2B Energy-optimised junction approach for public transport vehicles	MAP, SPAT Public transport vehicles can time their approach to junction in a way to save energy, by not having to stop and re-depart.
Stage 2B Time-optimised departure from stop for public transport vehicles	MAP, SPAT Public transport vehicles can keep their doors open longer to let passengers embark and disembark. The doors are closed at the right moment to catch the next green phase.
Stage 2B Navigation assistance	MAP, SPAT Public transport vehicles early instruction which lane to take, show layout of intersection, show signal phase and countdown to next phase

5 Protocol

5.1 Message Transport

The following guidelines apply for all clients:

Item	Guideline
Protocol	The application protocol is MQTT v5
Security	Security options: <ul style="list-style-type: none">• MQTT over TLS (port 8883) or MQTT over IP (port 1883)• server certificate for the MQTT broker: publicly verifiable or self-signed• method of client authentication: username+password or client certificate
Payload	the MQTT payload can be “binary” or “text”: <ul style="list-style-type: none">• binary: indicates that the message format is Protobuf• text: indicates that the message format is JSON (UTF-8) Notes: <ul style="list-style-type: none">• At least the initial prototyping will be using JSON only. If no performance issues will be found, a switch to protobuf, though currently not excluded, will never be required.• For air interface mirroring the payload will depend on the encoding of the message.

5.2 Message Encoding

The following guidelines apply for all messages:

Item	Guideline
Schema	For conciseness, the Protobuf schema has been used to specify the messages. A derived (generated) JSON schema representation will be provided. Note: Arrays of arrays cannot be represented in Protobuf and must therefore not be used, to allow a later migration to Protobuf, if necessary.
Header	There is a common header for each message, initially consisting of the following attributes: timestamp, version.
Versioning	The messages shall be extensible without creating any compatibility issues. Only as a very last resort – if compatibility cannot be achieved – shall the protocol “version” attribute (integer, starting at 1) be incremented. If this is the case, the version applies to all messages that are defined in the same proto file. The frame itself is not versioned.
Encoding	The messages shall be serialised as JSON, using the canonical mapping as defined in the following page: https://protobuf.dev/programming-guides/proto3/#json . Note the special rules for mapping the following non-primitive data types to JSON: <ul style="list-style-type: none">• google.protobuf.Timestamp → string (as per RFC 3339)• google.protobuf.Duration → string (number with fractional digits, followed by “s”)• google.protobuf.StringValue → string (nullable)• google.protobuf.FloatValue → number (nullable)• google.protobuf.DoubleValue → number (nullable)• google.protobuf.BoolValue → true false (nullable)• google.protobuf.Int32Value → number (nullable) The notation with offset (\pm HH:mm) in local wallclock time is preferred over UTC for timestamps.
Extensibility	The protobuf schema definitions shall have extensibility in mind. These best practices shall apply: https://protobuf.dev/programming-guides/dos-donts/ .

Item	Guideline
Optionality	<p>Optional attributes shall be included in the message if, and only if, the publisher knows the value:</p> <ul style="list-style-type: none"> • If the value is known to the publisher (depending on the situation), it <u>must</u> be provided. • If the value is unknown to the publisher, the attribute must not be provided. It is not permitted to provide the attribute with a value that is deemed outside of the valid range. <p>In Protobuf, unset attributes are set to default upon de-serialisation and hence cannot be distinguished from explicit default values. Therefore, optionality of an attribute is expressed by using a Protobuf wrapper instead of a primitive type. In the generated JSON schema, there is no wrapper and unset attributes are set to “undefined” upon de-serialisation.</p>
Requiredness	Requiredness of an attribute is expressed by marking the attribute with the JSON generator field option “required”. This property will be present in the JSON schema.
Maximum Size	The maximum size for incoming messages is XX MByte. Received messages that exceed this maximum size shall be discarded without parsing and without attempting to log the content.

5.3 Message Header

The message header is common to all messages. It is defined as follows:

Item	Definition
Structure (Protobuf 3)	<pre>message PtxHeader { google.protobuf.Timestamp timestamp = 1 [required = true]; // [RFC 3339, 3 decimals] time of sending int32 version = 2 [required = true]; // initial version = 1 } // the header is not versioned -- only compatible extensions!!</pre>

Attribute definitions:

Attribute	Definition
timestamp	The timestamp when the message was sent. Each party shall ensure that it is synchronized to UTC. It is permitted to use local time zones. The resolution shall be milliseconds.
version	The version determines the schema version of the message (“msg”). The version shall only be increased when incompatible changes are required. Preferably, the messages are extended in a compatible manner, not requiring the version to be incremented. As per this version of the specification, the value is 1.

5.4 Message Topics

Separate topics are defined for each message. For the <device type> part, the following strings shall be used:

- “v2x”: on-board unit (OBU) for V2X communication
- “ibis”: integrated on-board information system (aka. automatic vehicle location system)

The concrete <device> needs to be unique within the device type.

It is foreseeable that the topic structure will need to be changed when contributing this specification into VDV 435. Some flexibility (configurability) will hence be required until the topic structure can be considered “frozen”.

5.4.1 Topic Prefixes

In the description, the topics have been marked as “publisher-oriented” and “subscriber-oriented”. This is to be interpreted as follows:

Publishing Device	Publisher-oriented Topic Prefix	Subscriber-oriented Topic Prefix
V2X OBU	v2x/<v2x device>/...	ibis/<ibis device>/...
IBIS OBC	ibis/<ibis device>/...	v2x/<v2x device>/...

5.4.2 Topic Directory

The following table presents an overview of the defined topics and suggests access permissions. It is however up to the implementation to decide on such access permissions.

Prefix	Topic	Publisher	Subscriber	Bridge	IBIS	OBU	other
Device Management							
ibis/<ibis device>	power/state	only IBIS	any device	R	W	R	R
<device type>/<device>	log/level	only IBIS	specific device	W	W	R	R
<device type>/<device>	trigger	only IBIS	specific device	W	R	R	R
<device type>/<device>	power/request	any device	only IBIS	R	R	W	W
<device type>/<device>	log/<tag>	any device	only IBIS	R	R	W	W
<device type>/<device>	version	any device	only IBIS	R	W	W	W
<device type>/<device>	health	any device	only IBIS	R	W	W	W
Operational Information							
ibis/<ibis device>	vehicle/info	only IBIS	any device	R	W	R	R
ibis/<ibis device>	operation/logon	only IBIS	any device	R	W	R	R
ibis/<ibis device>	operation/journey	only IBIS	any device	R	W	R	R
ibis/<ibis device>	operation/status	only IBIS	any device	R	W	R	R
V2X-specific Messages							

Prefix	Topic	Publisher	Subscriber	Bridge	IBIS	OBUs	other
v2x/<v2x device>	config	only IBIS	only OBU	R	W	R	--
v2x/<v2x device>	path/definition	only IBIS	only OBU	R	W	R	--
v2x/<v2x device>	path/location	only IBIS	only OBU	R	W	R	--
v2x/<v2x device>	r09request/<intersection id>/<reporting point nr>	only IBIS	only OBU	R	W	R	--
v2x/<v2x device>	capabilities	only OBU	only IBIS	R	R	W	--
v2x/<v2x device>	r09response/<intersection id>/<reporting point nr>	only OBU	only IBIS	R	R	W	--
v2x/<v2x device>	intersection/<intersection id>/map	only OBU	only IBIS	R	R	W	--
v2x/<v2x device>	intersection/<intersection id>/phase	only OBU	only IBIS	R	R	W	--
v2x/<v2x device>	intersection/<intersection id>/status	only OBU	only IBIS	R	R	W	--
Air Interface Mirroring							
v2x/<v2x device>	out/<message type>/<encoding rule>	only OBU	only IBIS	R	R	W	--
v2x/<v2x device>	in/<message type>/<encoding rule>	only OBU	only IBIS	R	R	W	--

5.5 Message Expiration

To avoid garbage (e.g. MAP messages of intersections that no longer exist or have been reconfigured to use a different ID) to accumulate in the broker, all messages shall be published with an expiry of no more than 100 hours. Depending on the type of message, the expiry may be shorter. The message expiry feature is one of the reasons why MQTT v5 must be used.

The intended expiry duration is defined for each retained message.

5.6 Message Repetition

Some messages are to be repeated even if there is no change. The intended repetition rate is defined for each such message.

5.7 Message Attributes

The attributes of the MQTT “publish” messages shall be used as follows:

Attribute	Definition
topic	c.f. message definitions
payload	message content
qos	c.f. message definitions
retain	c.f. message definitions
expiry	c.f. message definitions (never >100 hours, should be configurable per message)
payload format indicator	1 (= UTF-8)
content type	application/json (or other MIME type if permitted as per §5.2)

6 Device Management and Logging

These messages are not application-specific. The overlap with messages that have been defined in the context of IBIS-IP (VDV301) is intended, in order to allow the V2X OBUs to use only the broker-based interface, without the need for IBIS-IP services.

6.1 Enumeration Types

6.1.1 Module Class

Value	Definition
CLASS_HW	Hardware module.
CLASS_FW	Firmware module.
CLASS_OS	Operating system module.
CLASS_SW	Application software module.
CLASS_DATA	Operational data supply.
CLASS_CFG	Configuration data (settings).

6.1.2 Reachability

Value	Definition
REACHABLE_DIRECT	The device reports its health directly to the broker.
REACHABLE_YES	The publisher acts as a proxy and can currently reach the device (and is therefore able to report the device's health to the broker).
REACHABLE_NO	The publisher acts as a proxy and cannot currently reach the device (and is therefore currently unable to report the device's health to the broker).

6.1.3 Activation

Value	Definition
STATUS_UNKNOWN	The activation status of the device is unknown because it is not reachable.
STATUS_ACTIVE	The device is active and shall be monitored.
STATUS_INACTIVE	The device is inactive. It will not send any updates for a while, but this is normal and shall not trigger any monitoring alerts.

6.1.4 Health

Value	Definition
HEALTH_UNKNOWN	The health of the device is unknown because it is not reachable.
HEALTH_OK	The device is fully operational and there is nothing to report.
HEALTH_INFO	The device reports an anomaly which does not impair its function. The details are provided in the "reason" attribute.
HEALTH_YELLOW	The device reports an anomaly which does impair its function. Whether or not this health status can be used in a meaningful way depends on the device. The details are provided in the "reason" attribute.
HEALTH_RED	The device is strongly impaired or even not available. The details are provided in the "reason" attribute.

6.1.5 Trigger

Value	Definition
TRIGGER_REBOOT	This command triggers the device to reboot. Needs not to be supported by all devices.
TRIGGER_PUBLISH	This command triggers the device to publish "something". The "args" attribute (which is application-specific) lets the device know how to interpret this trigger.

6.1.6 Log Level

Value	Definition
LEVEL_OFF	When used to control the log level, this log level nothing shall be logged. This level cannot be used in any log message.
LEVEL_FATAL	When used to control the log level, this log level indicates that only fatal errors (anomalies hindering the device from working) shall be logged. When used in a log message, this log level indicates that the reported anomaly is a fatal error (the device does not work as intended)
LEVEL_ERROR	When used to control the log level, this log level indicates that only errors and fatal errors (anomalies preventing normal execution) shall be logged. When used in a log message, this log level indicates that the reported anomaly is indeed an error (not all functions work fine).
LEVEL_WARNING	When used to control the log level, this log level indicates that errors and fatal errors (anomalies preventing normal execution) and warnings shall be logged. When used in a log message, this log level indicates that the reported anomaly is indeed a warning (something unexpected happened).
LEVEL_INFO	When used to control the log level, this log level indicates that errors and fatal errors (anomalies preventing normal execution), warnings and information messages shall be logged. When used in a log message, this log level indicates that the reported message is indeed just for information (significant to the purpose).

6.1.7 Power State

Value	Definition
POWER_ACTIVE	Power is normally available and there is currently no plan to switch it off.
SWITCH_OFF_PLANNED	Power is available and it is planned to switch it off at the indicated time. The peripheral devices can request an extension, e.g. to complete a data transfer.
SWITCH_OFF_IMMINENT	Power is available and it will be switched off at the indicated time. The peripheral devices can no longer request any extensions and must ensure that power off does not lead to any inconsistent states.

6.2 IBIS-to-Device Messages

6.2.1 Power State

Item	Definition
Name	PtxDmPowerState
Purpose	This information helps the OBU (and any other device) to decide on its own activation level
Pub Topic	ibis/<ibis device>/power/state [type and ID of publishing (= managing) device]
Structure (Protobuf 3)	<pre>message PtxDmPowerState { io.ptx.proto.PtxHeader msg_header = 1 [required = true]; PtxDmEnum.DmDevicePowerStateEnum power_state = 2 [required = true]; google.protobuf.BoolValue ignition_on = 3; google.protobuf.BoolValue comm_available = 4; google.protobuf.BoolValue bulk_available = 5; google.protobuf.Timestamp shutdown_not_before = 6; // [RFC 3339, no decimals] }</pre>
Retain	true (expiry: 20 minutes; frequency: every 10 minutes or upon change)
QoS	1
Behaviour of publisher (IBIS)	When the ignition is switched off, the devices should go to sleep after a while in order not to drain the battery. The IBIS will indicate that a switch-off is planned. The devices can request an extension. One minute (= grace period) before really switching off, it will indicate that a switch-off is imminent. The devices can now no longer request any extension. When the grace period has expired, the IBIS will switch off the devices by means of the LK2 signal.
Behaviour of Subscriber (OBU)	The OBU adapts its activation level in line with the power state. When the switch-off is imminent, it terminates any activities that might leave it in an inconsistent state, such as writing to permanent storage.

Attribute definitions:

Attribute	Definition
power state	The current power state indicated by the on-board computer.
ignition on	Indicates if the ignition is currently on (true) or off (false).
comm available	Indicates that communication is currently available (true) or unavailable (false). This usually means that a mobile network connection is available.
bulk available	Indicates that communication for bulk data transfer is currently available (true) or unavailable (false). This usually means that a WIFI connection is available. When available, bulk uploads and downloads are allowed without any limitations. When not available, bulk uploads or downloads should preferably be avoided. Whether or not they are allowed at all in this situation is subject to customer requirements and hence needs to be configurable.
shutdown not before	This timestamp indicates the earliest time when the power will be switched off. Depending on the power state, the device may request an extension. The extended time will be reflected in this attribute. This attribute must be provided if the power state is either SWITCH_OFF_PLANNED or SWITCH_OFF_IMMINENT. Otherwise, it is optional.

6.2.2 Log Level Configuration

Item	Definition
Name	PtxDmLogLevel
Purpose	This message tells the addressed device from which log level onwards to publish on the broker
Sub Topic	<device type>/<device>/log/level [type and ID of subscribing (= managed) device]
Structure (Protobuf 3)	<pre>message PtxDmLogLevel { io.ptx.proto.PtxHeader msg_header = 1 [required = true]; PtxDmEnum.DmDeviceLogLevelEnum Level = 2 [required = true]; }</pre>
Retain	true (expiry: 50 hours; frequency: every 24 hours or upon change)
QoS	1
Behaviour of OBU	The OBU publishes log messages as per the configured level (and higher).

Attribute definitions:

Attribute	Definition
log level	The desired log level that the device should use to publish log messages.

6.2.3 Command Trigger

Item	Definition
Name	PtxDmTrigger
Purpose	This message is used to trigger a command on the addressed device.
Sub Topic	<device type>/<device>/trigger [type and ID of subscribing (= managed) device]
Structure (Protobuf 3)	<pre>message PtxDmTrigger { io.ptx.proto.PtxHeader msg_header = 1 [required = true]; PtxDmEnum.DmDeviceTriggerEnum cmd = 2 [required = true]; repeated string args = 3; }</pre>
Retain	false (!!)
QoS	2
Behaviour of OBU	If supported, the command is to be executed immediately. In case of a reboot, the device shall first publish its health status with activation status _INACTIVE. This allows the system monitoring tool to avoid raising any alarms while the OBU is rebooting.

Attribute definitions:

Attribute	Definition
cmd	The command to be executed by the device.
args	Any arguments that the command may require, as a list of strings (application-specific).

6.3 Device-to-IBIS Messages

6.3.1 Power Request

Item	Definition
Name	PtxDmPowerRequest
Purpose	This information lets the OBU (or any other device) request power, e.g. during a firmware update.
Pub Topic	<device type>/<device>/power/request [type and ID of publishing (= managed) device]
Structure (Protobuf 3)	<pre>message PtxDmPowerRequest { io.ptx.proto.PtxHeader msg_header = 1 [required = true]; google.protobuf.Timestamp extension = 2 [required = true]; // [RFC 3339, no decimals] bool comm_request = 3 [required = true]; bool bulk_request = 4 [required = true]; }</pre>
Retain	false
QoS	2
Behaviour of IBIS	After the IBIS has indicated that the switch-off is planned, it will wait until none of the peripherals requests an extension or until all the requested extensions have expired or until a maximum time has passed (to save the vehicle battery in case of erroneous extensions). After that, the IBIS will indicate that the switch-off is imminent and will then switch off the LK2 signal. Whenever a request for extension is received, a new Power State message shall immediately be published, whether the information content has changed.
Behaviour of OBU	If the OBU performs operations that should not be interrupted, it shall request an extension for power and optionally for communication.

Attribute definitions:

Attribute	Definition
extension	Requesting an extension – of the grace period until the power is switched off – is optional. If an extension requested by means of this message, then the flags “comm request” and “bulk request” can be used to express the need for communication.
comm request	The device requests communication to be available until the power is switched off. If not provided, this flag defaults to false. If communication for bulk transfer is requested at the same time, this flag has no effect.
bulk request	The device requests communication for bulk transfer to be available until the power is switched off. If not provided, this flag defaults to false.

6.3.2 Log Messages

All on-board devices (including the OBU) shall publish their logs to the broker, as per the defined log level.

Item	Definition
Name	PtxDmLogMessage
Pub Topic	<device type>/<device>/log/<tag> [type and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxDmLogMessage { io.ptx.proto.PtxHeader msg_header = 1 [required = true]; google.protobuf.Timestamp timestamp = 2 [required = true]; // [RFC 3339, 3 decimals] PtxDmEnum.DmDeviceLogLevelEnum Level = 3 [required = true]; string tag = 4 [required = true]; string msg = 5 [required = true]; }</pre>
Retain	false
QoS	0
Behaviour of OBU	The OBU publishes log messages as per the configured level (and higher). The same tag is used as part of the topic and within the message.

Attribute definitions:

Attribute	Definition
timestamp	The timestamp when the log message was created. Each party shall ensure that it is synchronized to UTC. It is permitted to use local time zones. The resolution shall be milliseconds.
level	The log level to which this message pertains.
tag	The device can attach an arbitrary tag to the log message, to aid tracing certain features across log messages that are written by different modules.
msg	The log message is a clear text in English. The structure and content of the log message is intentionally not defined. Its purpose is to let field engineers understand what is going on. As an example, the field engineer could trigger a self-test of the OBU and then check the logs against what is documented in the engineering manual. This logging is explicitly not intended for log analysis by developers.

6.3.3 Device Version Info

All on-board devices (including the OBU) shall report their version information upon boot and whenever changed while in operation. Customers may require this information to be forwarded to a central network monitoring system.

Item	Definition
Name	PtxDmVersion
Pub Topic	<device type>/<device>/version [type and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxDmVersion { message DmModuleVersion { PtxDmEnum.DmDeviceModuleClassEnum class = 1 [required = true]; string name = 2 [required = true]; string version = 3 [required = true]; } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; string description = 2 [required = true]; repeated DmModuleVersion module = 3; }</pre>
Retain	true (expiry: 50 hours; frequency: every 24 hours or upon change)
QoS	1
Behaviour of Publisher	The "device version publisher" shall <ul style="list-style-type: none"> publish a message upon each change of version information. The publisher might act as a proxy for other devices and thus publish version information for more than one device type or device publish a complete message every time, including all known versions

Attribute definitions:

Attribute	Definition
description	The description shall allow a field engineer to find and identify the device in the vehicle. This description should be identical to the "description" field of the Device Health Info message.
module	A list of substructures. Each substructure pertains to an internal module for which the device reports version information.
module.class	The module class. The main categories are hardware, software, and data. These categories are further subdivided into module classes.
module.name	The name of the module. In most cases, the module type is sufficient. In case there are multiple modules of the same type, the name must allow them to be distinguished. It is recommended, but not required, to use the ":" sign if the module name is made up of different parts.
module.version	The version of the module. This is an arbitrary string that is defined by the device manufacturer.

6.3.4 Device Health Info

All on-board devices (including the OBU) shall periodically (every 5-30 secs) report their health. Customers may require this information to be forwarded to a central network monitoring system.

Item	Definition
Name	PtxDmHealth
Pub Topic	<device type>/<device>/health [type and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxDmHealth { message DmResourceUsage { float cpu = 1 [required = true]; // [0%..100%, max. 1 decimal] overall CPU Load (running average) float ram = 2 [required = true]; // [0%..100%, max. 1 decimal] usage of the RAM float disk = 3 [required = true]; // [0%..100%, max. 1 decimal] usage of <u>the</u> (exactly 1) monitored partition } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; string description = 2 [required = true]; PtxDmEnum.DmDeviceReachabilityEnum reachability = 3 [required = true]; PtxDmEnum.DmDeviceActivationEnum activation = 4 [required = true]; PtxDmEnum.DmDeviceHealthEnum health = 5 [required = true]; google.protobuf.StringValue reason = 6; DmResourceUsage usage = 7; google.protobuf.Int32Value uptime = 8; // [s] time since last boot }</pre>
Retain	true (expiry: 75 hours; frequency: every 5 minutes)
QoS	1
Behaviour of Publisher	<p>The "device health publisher" shall</p> <ul style="list-style-type: none"> publish a message upon each change of status. The publisher might act as a proxy for other devices and thus publish status information for more than one device type or device publish a complete message every time, including all known values at least description, health, and reason (if not healthy) are mandatory

Attribute definitions:

Attribute	Definition
description	The description shall allow a field engineer to find and identify the device in the vehicle. This description should be identical to the "description" field of the Device Version Info message.
reachability	In case the device reports its own health, the reachability is "direct". If the device acts as a proxy to report the health of a connected device, the reachability indicates whether or not the connected device is currently reachable. The reachability has an impact on the other attributes.
activation	A device can indicate whether or not it is currently active. If the device is active, the device must publish its health in regular intervals. If the device is inactive, it needs not publish its health. Before becoming inactive (sleep or shutdown), the device shall publish a health message with an "inactive" state. When the device is inactive, the system monitoring tool needs not create any alerts for this device. If a proxied device cannot be reached, its activation is "unknown".
health	This is the main health status of the device. It must always be provided, even if detailed usage information is included in the message.
reason	If the health status is not OK, the reason must be given in a plain English message that allows a field engineer to understand the problem.
usage	A single substructure. The substructure is optional. If provided, it groups usage information for its most important resources: CPU, RAM, disk.
usage .cpu	The current load (non-idle time) of the CPU (all cores combined) in % (max. 1 decimal).

Attribute	Definition
usage.ram	The current usage of the RAM in % (max. 1 decimal).
usage.disk	The current usage of the monitored disk partition in % (max. 1 decimal). Which (of several) partitions to monitor is up to the implementaiton. Typically, only the partition where data or log files are written needs to be monitored, whereas OS and application partitions need not be monitored.
uptime	The uptime of the device in seconds (no decimals).

7 Operational Information

These messages are specific to public transportation, but not specific to the integration of V2X OBUs. The overlap with messages that have been defined in the context of IBIS-IP (VDV301) is intended, in order to allow the V2X OBUs to use only the broker-based interface, without the need for IBIS-IP services.

7.1 Enumeration Types

7.1.1 Vehicle Category

Value	Definition
CAT_OTHER	The category of the vehicle is unknown, or it doesn't fit into any of the other categories.
CAT_BUS	The vehicle is a bus. This includes any kind of bus: trolley or regular, articulated or standard, double or single decker, rapid or slow, ...
CAT_TROLLEY	The vehicle is a trolley bus or a bus that is capable of in-motion charging. This includes buses that can cover some parts of their journey on battery only.
CAT_TRAM	The vehicle is a tram. This means that it shares the space with road vehicles at least on some stretches of rail and may use road intersections.
CAT_RAIL	The vehicle is a rail vehicle that never shares the space with road vehicles. This includes any kind of rail vehicle with the exception of trams: light or heavy rail, subway, magnetic rail, ...
CAT_FUNI	The vehicle is a rail shuttle on a dedicated linear track. V2X may be used at terminals or alongside the track cable.
CAT_GONDOLA	The vehicle is a suspended shuttle on a dedicated linear track. V2X may be used at terminals or alongside the suspension cable.
CAT_FERRY	The vehicle is a ferryboat or other water bound vehicle. V2X may be used at terminals.

7.1.2 Location Status

Value	Definition
LOC_NONE	The vehicle is not currently logged onto a journey or pattern. Hence, there is no logical location.
LOC_OFF_COURSE	The vehicle is logged onto a journey or pattern but is not on-course. This situation is called off-course.
LOC_ON_COURSE	The vehicle is logged onto a journey or pattern. It is either located at a stop or between stops and on the intended path between the previous and the next stop. This situation is called on-course.

7.1.3 Priority Level

Value	Definition
PRIOR_NORMAL	When approaching an intersection, if the capabilities allow, traffic light priority shall be requested.
PRIOR_OFF_AT_STOP	The vehicle is currently located at a stop, and it might take a longer break. Despite being in commercial operation, no traffic light priority shall be requested.
PRIOR_OFF	The vehicle is currently not in commercial operation. No traffic light priority shall be requested.

7.1.4 Driver Cab Activation

Value	Definition
CAB_NONE	No driver cab is currently active.
CAB_A	The driver cab A is currently active.
CAB_B	The driver cab B is currently active.

7.2 IBIS-to-Device Messages

Operational information is published by the IBIS on-board computer. It is addressed “to whom it may concern”, including the OBU.

7.2.1 Vehicle Info

This message lets the IBIS provide information about the vehicle to the OBU, so that the latter knows the static and low-frequency information pertaining to the “regular” CAM (without R09 payload). Note that the dimensions of a vehicle can change due to coupling (e.g. a trailer).

Item	Definition
Name	PtxOIVehicleInfo
Pub Topic	ibis/<ibis device>/vehicle/info [type (always “ibis”) and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxOIVehicleInfo { io.ptx.proto.PtxHeader msg_header = 1 [required = true]; PtxOIVehicleCategory category = 2 [required = true]; google.protobuf.StringValue type = 3; // for debugging only, non-null and non-empty if provided google.protobuf.BoolValue has_trailer = 4; google.protobuf.Int32Value nof_vehicles = 5; // >1 indicates coupling (tram or train) google.protobuf.FloatValue weight = 6; // [kg, 0 decimals] sum for whole train google.protobuf.FloatValue length = 7; // [m, max. 2 decimals] sum for whole train google.protobuf.FloatValue width = 8; // [m, max. 2 decimals] max for whole train google.protobuf.FloatValue height = 9; // [m, max. 2 decimals] max for whole train google.protobuf.Int32Value capacity = 10; // [pax] sum for whole train google.protobuf.StringValue plate = 11; // non-null and non-empty if provided google.protobuf.StringValue vin = 12; // non-null and non-empty if provided }</pre>
Retain	true (expiry: 50 hours; frequency: every 24 hours or upon change)
QoS	1
Behaviour of IBIS	The IBIS shall provide the vehicle info at startup and only upon change. This may happen during operation, when trailers or vehicles are coupled.

Attribute definitions:

Attribute	Definition
category	The vehicle category. Compared to the vehicle types defined in V2X, the list is extended to cover also other public transport vehicle types which are commonly used, and which may benefit from V2X technology.
type	The vehicle type as a string. This is for debugging only.
has_trailer	If true, this vehicle has a trailer. Typically, trailers are used for buses.
nof_vehicles	if >1, multiple vehicles have been coupled together to form a train. This could also be a road train, consisting of vehicles operated in convoy mode. Each vehicle is assumed to have an IBIS on-board computer and one or two V2X on-board unit(s), but only the IBIS on-board computer in the leading vehicle is active. The activation of the OBUs depend on the “active driver cab” information.
weight	The vehicle’s (or train’s) <u>empty</u> weight in kilograms. In case of a train, this is the sum of all coupled vehicles. The empty weight might influence the vehicle’s priority at the intersection.
length	The vehicle’s (or train’s) overall length in meters. In case of a train, this is the sum of all coupled vehicles.
width	The vehicle’s (or train’s) width in meters. In case of a train, this is the maximum width of all coupled vehicles.

Attribute	Definition
height	The vehicle's (or train's) height in meters. In case of a train, this is the maximum height of all coupled vehicles.
capacity	The vehicle's (or train's) overall passenger capacity in pax. In case of a train, this is the sum of all coupled vehicles.
plate	(for road vehicles only) The vehicle's licence plate.
vin	(for road vehicles only) The globally unique "vehicle identification number".

7.2.2 Operational Logon

The operational logon message combines various attributes that relate to the vehicle's current operational logon (and thereby to the task that the vehicle is executing). Its purpose is to convey the operational (logical) logon in a similar way as today's VDV-300 (IBIS) vehicle bus.

To be meaningful, IDs that jointly comprise the logon information must be linked back to the data supply (of network and timetable data).

Item	Definition
Name	PtxOOperationalLogon
Pub Topic	ibis/<ibis device>/operation/logon [type (always "ibis") and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxOOperationalLogon { // IDs refer to VDV452 and VDV455B io.ptx.proto.PtxHeader msg_header = 1 [required = true]; string vehicle_id = 2 [required = true]; // VDV452/443 VEHICLE(FAHRZEUG)::VEHICLE_NO(FZG_NR) google.protobuf.StringValue driver_id = 3; // VDV455B DRIVER(PERSONAL)::DRIVER_NR(FAHRER_NR) google.protobuf.StringValue block_id = 4; // VDV452/310 BLOCK::BLOCK_NO(UM_UID) google.protobuf.StringValue journey_id = 5; // VDV452/715 JOURNEY(REC_FRT)::JOURNEY_NO(FRT_FID) google.protobuf.StringValue pattern_id = 6; // VDV452/226 LINE(REC_LID)::ROUTE_NO(ROUTEN_NR) google.protobuf.StringValue line_id = 7; // VDV452/226 LINE(REC_LID)::LINE_NO(LI_NR) google.protobuf.StringValue direction_id = 8; // VDV452/226 LINE(REC_LID)::DIRECTION(LI_RI_NR) google.protobuf.StringValue run_id = 9; // VDV452/715 JOURNEY(REC_FRT)::RUN(LT_KU_NR) google.protobuf.StringValue daytype_id = 10; // VDV452/290 DAY_TYPE(MENGE_TAGESART)::DAY_TYPE_NO(TAGESART_NR) google.protobuf.StringValue opday_id = 11; // VDV452/348 PERIOD(FIRMENKALENDER)::OPERATING_DAY(BETRIEBSSTAG) }</pre>
Retain	true (expiry: 20 minutes; frequency: every 10 minutes or upon change)
QoS	1
Behaviour of Publisher	<p>The following applies:</p> <ul style="list-style-type: none"> all currently known values shall be provided in every message (apart from driver ID which may be restricted) the message shall be republished upon every change of an attribute (typically at the start of every journey, typically every 30..60 minutes)

Attribute definitions:

Attribute	Definition
vehicle id	The ID of the vehicle. This ID is not the same as the VIN (vehicle identification number) which is used in different contexts as the globally unique vehicle ID. Ref.: VDV452/443 vehicle(fahrzeug)::vehicle_no(fzg_nr)
driver id	The ID of the driver. The driver ID might not always be available due to data protection regulations. Ref.: VDV455B driver(personal)::driver_nr(fahrer_nr)
block id	The ID of the block (= sequence of journeys). Ref.: VDV452/310 block::block_no(um_uid)
journey id	The ID of the journey (= sequence of calls). Ref.: VDV452/715 journey(rec_frt)::journey_no(frt_fid)
pattern id	The ID of the pattern (= sequence of stop points) underlying the journey. Ref.: VDV452/226 line(rec_lid)::route_no(routen_nr)
line id	The ID of the line (= grouping of patterns). Ref.: VDV452/226 line(rec_lid)::line_no(li_nr)
direction id	The ID of the direction (= grouping of patterns within a line). Ref.: VDV452/226 line(rec_lid)::direction(li_ri_nr)
run id	The ID of the run (= together with the line, identifier of a block). Ref.: VDV452/715 journey(rec_frt)::run(li_ku_nr)

7.2.3 Operational Journey

The operational journey message provides information pertaining to the journey that the vehicle is currently executing. Its purpose is to convey the journey in a similar way as today's VDV-300 (IBIS) vehicle bus.

To be meaningful, IDs that jointly comprise the logon information must be linked back to the data supply (of network and timetable data).

Item	Definition
Name	PtxOiOperationalJourney
Pub Topic	ibis/<ibis device>/operation/journey [type (always "ibis") and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxOiOperationalJourney { // IDs refer to VDV452 message OiJourneyCall { message OiStopPoint { string id = 1 [required = true]; // if planned stop point: VDV452/253 STOP(REC_ORT)::POINT_NO(ORT_NR) string name = 2 [required = true]; // stop short name + "-" + point nr (for debugging purposes) google.protobuf.DoubleValue lat = 3; // [deg WGS-84, -90..+90, pos. = N] mandatory if available google.protobuf.DoubleValue lon = 4; // [deg WGS-84, -180..+180, pos. = E] mandatory if available google.protobuf.FloatValue heading = 5; // [deg 0..360, 0 = true north, 90 = east] mandatory if available } message OiCallData { google.protobuf.Timestamp timestamp = 1 [required = true]; // [RFC 3339, no decimals] } int32 call_seq = 1 [required = true]; // #[, 1-based] OiStopPoint stop_point = 2 [required = true]; // stop point at which the vehicle calls OiCallData arrival_data = 3; // information pertaining to the arrival OiCallData departure_data = 4; // information pertaining to the departure google.protobuf.FloatValue dist_to_next_stop = 5; // [m, max. 1 decimal] google.protobuf.BoolValue do_not_dwell = 6; // indication that vehicle must continue even if ahead of time google.protobuf.Int32Value typical_dwell_time = 7; // [s] } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; google.protobuf.StringValue journey_id = 2 [required = true]; // VDV452/715 JOURNEY(REC_FRT)::JOURNEY_NO(FRT_FID) repeated OiJourneyCall call = 3; // ordered list of scheduled calls of this journey } </pre>
Retain	true (expiry: 20 minutes; frequency: every 10 minutes or upon change)
QoS	1
Behaviour of Publisher	<p>The following applies:</p> <ul style="list-style-type: none"> the complete scheduled journey shall be provided upon logoff from the journey, an empty message shall be published to clear it

Attribute definitions:

Attribute	Definition
journey_id	The ID of the journey (= sequence of calls). Ref.: VDV452/715 journey(rec_frt)::journey_no(frt_fid)
call	A list of substructures. Each substructure pertains to a call (= visit at a stop point) of the journey.
call .call_seq	The 1-based sequence number of the call within its journey. This sequence number is referred to from the logical location.
call .stop_point	A single substructure denoting the stop point where the vehicle will call along its journey. The stop point coordinates will in general not be very precise. It is furthermore important to consider that the vehicle will in general not stop precisely at this stop point. Not only will there be a lateral difference. The longitudinal distance will typically be greater. In case of long platforms, vehicles may stop behind each other.

Attribute	Definition
call .stop point .id	The ID of the stop point at which the vehicle calls. Ref.: VDV452/253 stop(rec_ort)::point_no(ort_nr) Note: use a non-colliding ID for ad-hoc stop points (that are not part of the planning data)
call .stop point .name	The name of the stop point as a string. This name is provided for debugging purposes. It is typically constructed from the stop short name and the stop point number within the stop.
call .stop point .lat	The latitude of the stop point in degrees WGS-84, -90..+90, positive = north (6 or 7 decimals).
call .stop point .lon	The longitude of the stop point in degrees WGS-84, -180..+180, positive = east (6 or 7 decimals).
call .stop point .heading	The heading of the stop point in degrees, 0..360, 90 = east (0..1 decimals). The heading will in general be quite accurate, but this is not guaranteed. In some cases, the heading has been observed to be pointing 180° into the wrong direction.
call .arrival data	A single substructure pertaining to the arrival of the vehicle at the stop point.
call .arrival data .timestamp	The timestamp when the vehicle is scheduled to arrive at the stop, after any time offsets that result from a dispatch action have been applied.
call .departure data	A single substructure pertaining to the departure of the vehicle from the stop point.
call .departure data .timestamp	The timestamp when the vehicle is scheduled to depart from the stop, after any time offsets that result from a dispatch action have been applied.

7.2.4 Operational Status

The operational progress message combines various attributes that relate to the vehicle's current progress within the operational logon (the location within the vehicle's task). Its purpose is to convey the operational (logical) progress in a similar way as today's VDV-300 (IBIS) vehicle bus.

The operational status message should contain all high-frequency information that is needed to broadcast the “regular” CAM (without R09 payload).

Item	Definition
Name	PtxOiOperationalStatus
Pub Topic	ibis/<ibis device>/operation/status [type (always “ibis”) and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxOiOperationalStatus { message OiVehicleSignals { google.protobuf.BoolValue reverse_gear = 1; google.protobuf.BoolValue doors_released = 2; google.protobuf.BoolValue doors_open = 3; google.protobuf.BoolValue stop_brake_active = 4; } message OiGeoLocation { double latitude = 1 [required = true]; // [deg WGS-84, -90..+90, pos. = N, 6 decimals] double longitude = 2 [required = true]; // [deg WGS-84, -180..+180, pos. = E, 6 decimals] google.protobuf.FloatValue accuracy = 3; // [m, non-neg., max. 1 decimal] google.protobuf.FloatValue altitude = 4; // [m, pos. = up, 1 decimal] google.protobuf.FloatValue vertical_accuracy = 5; // [m, non-neg., max. 1 decimal] google.protobuf.FloatValue heading = 6; // [deg 0..360, 0 = true north, 90 = east, max. 1 decimal] google.protobuf.FloatValue speed = 7; // [m/s, max. 2 decimals] speed from GNSS } message OiLogicalLocation { string journey_id = 1 [required = true]; // reference to journey - reject logical location if not matching int32 call_seq = 2 [required = true]; // #[, 1-based] prev-or-current-or-next stop google.protobuf.FloatValue distance = 3; // [m, max. 1 decimal] neg. = before stop, pos. = after stop } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; PtxOiEnum.OiDriverCabActivation driver_cab_active = 2 [required = true]; OiVehicleSignals vehicle_signals = 3; google.protobuf.FloatValue odo_speed = 4; // [m/s] google.protobuf.Int32Value sat_count = 5; // #[# OiGeoLocation geo_loc = 6; PtxOiEnum.OiLocationstatus status = 7 [required = true]; OiLogicalLocation logical_loc = 8; // only if logged onto a journey google.protobuf.Int32Value deviation = 10; // [s, pos. = delayed] google.protobuf.FloatValue occupancy = 11; // [%], 0..100, max. 1 decimal PtxOiEnum.OiPriorityLevel prio_level = 12 [required = true]; }</pre>
Retain	false (frequency: every 0.5..2 seconds)
QoS	0
Behaviour of IBIS	
Behaviour of OBU	<p>Even if the OBU may have its own built-in GNSS module, it might be useful that the IBIS provide the geo location:</p> <ul style="list-style-type: none"> • might use dead-reckoning or other location-enhancing mechanisms • modified depending on the active driver cab for bi-directional vehicles <p>The embarkation status for the CAM message can be derived by OR-ing the flags doors_released, doors_open and stop_brake_active. If this result is “false”, the vehicle is ready to depart (or is already driving).</p>

Attribute definitions:

Attribute	Definition
driver cab active	<p>Indication of the active driver cab. Uni-directional vehicles have only cab "A". This information especially important for bi-directional vehicles:</p> <ul style="list-style-type: none"> the precise geo location as well as speed and heading of the vehicle are expressed relative to the active driver cab ("the <u>current front</u> of the vehicle") only the OBU associated with the active driver cab is allowed to be active over-the-air and to send any V2X-specific OBU-to-IBIS messages (c.f. chapter 8.3) to the IBIS <p>If there are multiple OBUs in a vehicle, the OBUs shall "activate" and "deactivate" themselves based on this attribute. At most 1 OBU per vehicle can be active at any given moment.</p>
vehicle signals	A single substructure. The substructure groups the binary signals that the IBIS obtains from the vehicle and republishes in this data structure.
vehicle signals .reverse gear	If true, the vehicle is moving backward with respect to the active cabin.
vehicle signals .doors released	If true, this flag indicates that the doors are released, i.e. that pushing a door button will result in the door opening.
vehicle signals .doors open	If true, this flag indicates that at least one door is currently open.
vehicle signals .stop brake active	If true, this flag indicates that the stop brake is currently activated. The vehicle is thus immobilised.
odo speed	<p>The speed in meters per second that is derived from the odometer. This speed should be available even in the case where the GNSS speed is not, e.g. in a tunnel.</p> <p>If the odo speed is unknown, it shall either not be provided or then set to 0.</p> <p>Note: If the vehicle is in reverse gear and moving, the odo speed is negative. This means that the speed calculated from the odometer must be negated.</p>
sat count	<p>The number of satellites that contributed to the geo location. This value complements the accuracy estimation.</p> <p>Note that if dead-reckoning (sensor fusion) is used by the GNSS, it may provide a good geo location even in the absence of any visible satellites.</p>
geo loc	<p>A single substructure. The substructure groups the information that has been obtained from the GNSS module. If the coordinates are not valid ("no fix"), this structure shall not be provided at all.</p> <p>Note: In case of a bi-directional vehicle with a single GNSS antenna, the latitude and longitude must be adjusted to the active cabin before publishing the geo location.</p>
geo loc .latitude	The latitude of the front of the vehicle in degrees WGS-84, -90..+90, positive = north (6 or 7 decimals).
geo loc .longitude	The longitude of the front of the vehicle in degrees WGS-84, -180..+180, positive = east (6 or 7 decimals).
geo loc .accuracy	Distance in meters (0..1 decimal). With 95% probability, the vehicle is located no more than this distance away from the point defined by latitude and longitude.
geo loc .altitude	<p>The altitude in meters above the WGS-84 ellipsoid (0..1 decimals).</p> <p>If the altitude is unknown, it shall not be provided.</p>
geo loc .vertical accuracy	<p>Difference in meters (0..1 decimals). With 95% probability, the vehicle's altitude is no more than this difference higher or lower.</p> <p>If the altitude is unknown, its accuracy is irrelevant.</p>

Attribute	Definition
geo loc .heading	<p>The direction in which the vehicle's front (active driver cab) is pointing in degrees (0..360, 90 = east, 0..1 decimals).</p> <p>If the heading is unknown, it shall either not be provided or then set to -1.</p> <p>Note:</p> <ul style="list-style-type: none"> • The vehicle might be in reverse gear and going backward (negative speed). The heading must in this case still express the orientation of the vehicle. This means that the heading obtained by the GNSS must be flipped by 180 degrees (plus 180 if less than 180 else minus 180). • The heading shall reflect the horizontal orientation of the first rigid part of the vehicle's body.
geo loc .speed	<p>The speed at which the vehicle's front is moving (speed of front wheels) in the direction of the active driver cab, in meters per second (0..2 decimals).</p> <p>If the speed is unknown, it shall either not be provided or then set to 0.</p> <p>Note: If the vehicle is in reverse gear and moving, the speed is negative. This means that the speed obtained from the GNSS must be negated.</p>
loc status	The status of the logical location.
logical loc	A single substructure. The substructure groups the information that is commonly known as "logical location". It is only available if the status of the logical location is either on-course or off-course, i.e. if the vehicle is logged onto a journey or pattern.
logical loc .journey id	The ID of the current journey. If this ID doesn't match (probably because of a race condition), then the logical location must be rejected, but the other elements of the operational status message remain valid.
logical loc .call seq	<p>The 1-based sequence number of a call within the current journey or pattern:</p> <ul style="list-style-type: none"> • If the vehicle is currently located at a stop, this is the current stop. • If the vehicle has left the stop and is on- or off-course to the next stop, this is the previous stop. • If the vehicle is heading towards this stop (it might be the first stop or the vehicle might have been instructed to visit this stop out of sequence), then this is the next stop.
logical loc .distance	<p>The distance in meters with respect to the referenced stop:</p> <ul style="list-style-type: none"> • If the vehicle is currently located at a stop, this distance is 0. • If the vehicle has left the stop and is on- or off-course to the next stop, this is the distance driven since the previous stop. • If the vehicle is heading towards this stop (... c.f. above ...) then this is the direct air distance to the next stop.
deviation	<p>The current timetable deviation of the vehicle in seconds. Positive = delayed, negative = ahead of schedule.</p> <p>While the vehicle is located at a stop, this is the expected deviation when departing from the stop.</p> <p>While the vehicle is located between stops (on- or off-course), this is the expected deviation when arriving at the next stop.</p> <p>When the vehicle is not logged onto a journey or pattern (unproductive operation), there is no deviation. It is then safe to use the 0 value.</p> <p>The deviation must be provided if the vehicle is currently on-course, and may be provided if the vehicle is currently off-course.</p> <p>The timetable deviation might influence the priority that is given to the vehicle by the traffic controller.</p>
occupancy	<p>The occupancy of the vehicle in percent (0..100).</p> <p>If the occupancy is unknown, it shall either not be provided or then set to -1.</p> <p>The occupancy might influence the priority that is given to the vehicle by the traffic controller.</p>

Attribute	Definition
prio level	<p>The priority level is used to tell the OBU whether traffic signal priority is currently requested when approaching a traffic signal. It will usually be set to "normal" when the vehicle is in commercial operation and to "off" when the vehicle is not in commercial operation. To deal with special cases that may occur while the vehicle is stationary at a stop for an extended period of time while in commercial operation, the status "off at stop" shall be used to indicate that no traffic signal priority requests shall be made at this moment.</p> <p>This "on/off switch" is to be ignored for R09 requests, which shall always be sent out, irrespective of this value.</p>

8 V2X-specific Messages

These messages are specific to the integration of V2X OBUs into an on-board system for public transportation.

8.1 Enumeration Types

8.1.1 Encoding Rule

Value	Definition
ENCODING_TEXT	V2X message provided as human-readable text (without any canonical conversion). As an example, the coordinates in the MAP message could be converted from the x/y format to WGS-84 lat/lion (e.g. "47.123456, 8.765432", for direct use in OpenStreetMap or Google Maps)
ENCODING_UPER	V2X message provided as binary (ASN.1 packed encoding rule).
ENCODING_JSON	V2X message provided as text (ASN.1 JSON encoding rule).
ENCODING_XML	V2X message provided as text (ASN.1 XML encoding rule).
ENCODING_PCAP	V2X message provided in PCAP format (e.g. for viewing with Wireshark).

8.1.2 Service Type

Value	Definition
SERVICE_R09_REQUEST_ONLY	The OBU can send out pre-serialised R09.16 telegrams embedded in a CAM. There is no response from the intersection.
SERVICE_R09_REQUEST_RESPONSE	The OBU can send out pre-serialised R09.16 telegrams embedded in an SRM. There might be a response from the intersection in the form of an SSM.
SERVICE_PHASE	The OBU understands the path that has been registered by the IBIS. For the relevant intersections, it provides intersection map and signal phase information to the IBIS.
SERVICE_PRIORITY	Building on the "phase" service, the OBU can request priority for the correct signal group of the upcoming intersection(s). It provides lane and priority status information back to the IBIS.
SERVICE_MAKE_AWARE	The OBU can send awareness information (= CAM message) about the ego vehicle.

8.1.3 Message Type

Value	Definition
MESSAGE_CAM	Co-operative awareness extended message. Refs: ETSI EN 302 637-2 / ISO TS 19091
MESSAGE_MAP	Intersection map. Refs: ETSI TS 103 301 / SAE J2735 / ISO TS 19091
MESSAGE_SPAT	Signal phase and timing. Refs: ETSI TS 103 301 / SAE J2735 / ISO TS 19091
MESSAGE_SRM	Signal request. Refs: ETSI TS 103 301 / SAE J2735 / ISO TS 19091
MESSAGE_SSM	Signal status. Refs: ETSI TS 103 301 / SAE J2735 / ISO TS 19091

8.1.4 Movement Phase State

Value	Definition
PHASE_UNAVAIL-ABLE	The state is unknown, or state information cannot currently be provided. Ref: J2735 – MovmementPhaseState “unavailable (0)”
PHASE_DARK	The signal is dark (unlit). Ref: J2735 – MovmementPhaseState “dark (1)”
PHASE_FLASHING_RED	The vehicle must stop but may proceed when it is safe (e.g. “turn on red”). Ref: J2735 – MovmementPhaseState “stop-then-proceed (2)”
PHASE_RED	The vehicle must stop and remain. Ref: J2735 – MovmementPhaseState “stop-and-remain (3)”
PHASE_RED_AND_YELLOW	The vehicle may prepare to move but must remain stopped. Ref: J2735 – MovmementPhaseState “pre-movement (4)”
PHASE_GREEN	The vehicle may proceed but may have to yield to conflicting traffic. Ref: J2735 – MovmementPhaseState “permissive-movement-allowed (5)”
PHASE_GREEN_EXCLUSIVE	The vehicle may proceed and has right of way in the indicated direction(s). Ref: J2735 – MovmementPhaseState “protected-movement-allowed (6)”
PHASE_YELLOW	The vehicle should stop if possible. If unable to stop, the vehicle may proceed but may have to yield to conflicting traffic. Ref: J2735 – MovmementPhaseState “permissive-clearance (7)”
PHASE_YELLOW_EXCLUSIVE	The vehicle should stop if possible. If unable to stop, the vehicle may proceed and has right of way in the indicated direction(s). Ref: J2735 – MovmementPhaseState “protected-clearance (8)”
PHASE_FLASHING_YELLOW	The vehicle may proceed with caution but may have to yield to conflicting traffic. The intersection controller might be offline for a while. Ref: J2735 – MovmementPhaseState “caution-conflicting-traffic (9)”

8.1.5 Priority Status

Value	Definition
STATUS_UNKNOWN	The status of the traffic signal is unknown. Ref: J2735 – PrioritizationResponseStatus “unknown (0)”
STATUS_REQUESTED	The traffic controller has received the request (= transient acknowledgement). Ref: J2735 – PrioritizationResponseStatus “requested (1)”
STATUS_PROCESSING	The traffic controller is processing the request (= transient acknowledgement). Ref: J2735 – PrioritizationResponseStatus “processing (2)”
STATUS_TRAFFIC	The traffic controller cannot give full permission – watch traffic (= final state). Ref: J2735 – PrioritizationResponseStatus “watch other traffic (3)”
STATUS_GRANTED	The traffic controller has granted permission (= final state). Ref: J2735 – PrioritizationResponseStatus “granted (4)”
STATUS_REJECTED	The traffic controller has rejected the request (= final state). Ref: J2735 – PrioritizationResponseStatus “rejected (5)”
STATUS_MAX	The traffic controller has expired the request (= final state). Ref: J2735 – PrioritizationResponseStatus “max presence (6)”
STATUS_LOCKED	The traffic controller currently doesn't accept any requests (= final state). Ref: J2735 – PrioritizationResponseStatus “service locked (7)”
STATUS_TIMEOUT	Timeout – the traffic controller did not respond within a reasonable time. (not a status present in J2735)

8.2 IBIS-to-OBU Messages

The messages in this section are sent from the IBIS to the OBU.

8.2.1 Configuration

This message lets the IBIS configure the OBU.

Item	Definition
Name	PtxV2xConfiguration
Sub Topic	v2x/<v2x device>/config [type (always “v2x”) and ID of subscribing device]
Structure (Protobuf 3)	<pre>message PtxV2xConfiguration { message V2xServiceConfig { PtxV2xEnum.V2xServiceType type = 1 [required = true]; // service to be activated int32 interval = 2 [required = true]; // [s] min. interval (per station) for OBU-to-IBIS messages } message V2xMessageConfig { PtxV2xEnum.V2xMessageType type = 1 [required = true]; // mirror messages of this type int32 interval = 2 [required = true]; // [s] min. interval per message of this type } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; repeated V2xServiceConfig service = 2; // services to be activated, can be null or empty repeated V2xMessageConfig incoming_msg = 3; // mirrored incoming messages, can be null or empty repeated V2xMessageConfig outgoing_msg = 4; // mirrored outgoing messages, can be null or empty PtxV2xEnum.V2xEncodingRule selected_rule = 5; // encoding rule for mirroring }</pre>
Retain	true (expiry: 50 hours; frequency: every 24 hours or upon change)
QoS	1
Behaviour of IBIS	The configuration sent by the IBIS must be in line with the capabilities that the OBU has reported. It is therefore sent as a response to a received capabilities message.
Behaviour of OBU	<p>Upon receiving the configuration message, the OBU shall</p> <ul style="list-style-type: none"> enable the specified services. The specified interval affects only the incoming messages, e.g. how often MAP or SPAT information is updated by publishing “IntersectionMap” or “IntersectionPhase” to the broker. mirror the messages sent to and received from the air as per the mirror configuration for incoming and outgoing messages. The specified interval affects only the frequency of mirroring, not the frequency in the air. <p>The configuration limits the rate at which incoming messages or services are published by the OBU to the broker, in order not to overwhelm the broker or the IBIS. The OBU will hence only down sample the received messages, but never up sample.</p>

Attribute definitions:

Attribute	Definition
service	A list of substructures. Each substructure pertains to a service in the sense of this specification. The IBIS shall only request services to be enabled that are supported by the OBU as per its capabilities.
service.type	The service which shall be enabled. If a service is not enabled by including it in this list, no incoming messages pertaining to this service shall be published to the broker by the OBU.

Attribute	Definition
service .interval	<p>The interval (per peer station) at which to forward incoming messages to the IBIS.</p> <p>An incoming message shall only be forwarded by the OBU to the IBIS if at least <interval> seconds have elapsed since the last message of the same type (e.g. "Intersection Map") has been forwarded for the same peer (e.g. the same RSU).</p>
incoming msg	<p>A list of substructures. Each substructure pertains to an incoming message type in the sense of the V2X (C-ITS) specification.</p> <p>The IBIS shall only request messages to be mirrored that are supported by the OBU as per its capabilities.</p>
incoming msg .type	<p>The incoming message type which shall be enabled for mirroring.</p> <p>If mirroring for a message type is not enabled by including it in this list, the incoming message shall not be mirrored to the broker by the OBU.</p>
incoming msg .interval	<p>The interval (per peer station) at which to mirror incoming messages to the broker.</p> <p>An incoming message shall only be mirrored by the IBIS to the broker if at least <interval> seconds have elapsed since the last message of the same type (e.g. SPAT) has been mirrored for the same peer (e.g. the same RSU).</p>
outgoing msg	<p>A list of substructures. Each substructure pertains to an outgoing message type in the sense of the V2X (C-ITS) specification.</p> <p>The IBIS shall only request messages to be mirrored that are supported by the OBU as per its capabilities.</p>
outgoing msg .type	<p>The outgoing message type which shall be enabled for mirroring.</p> <p>If mirroring for a message type is not enabled by including it in this list, the outgoing message shall not be mirrored to the broker by the OBU.</p>
outgoing msg .interval	<p>The interval (per peer station) at which to mirror outgoing messages to the broker.</p> <p>An outgoing message shall only be mirrored by the IBIS to the broker if at least <interval> seconds have elapsed since the last message of the same type (e.g. CAM) has been mirrored.</p> <p>The interval affects only the mirroring feature, not how often the message is sent out over the air. The latter is completely up to the OBU.</p>
selected rule	<p>For mirroring to the broker, the IBIS can select (among the rules supported by the OBU) which encoding rule to use.</p> <p>Note: PCAP will never be used for air interface mirroring. If PCAP capability is advertised by the OBU, then it will support the "trigger" mechanism to request the PCAP dumps to be published.</p>

8.2.2 Path Definition

This message lets the IBIS register and update a path to the OBU, so that the latter can correctly filter the MAP and SPAT information.

Item	Definition
Name	PtxV2xPathDefinition
Sub Topic	v2x/<v2x device>/path/definition [type (always "v2x") and ID of subscribing device]
Structure (Protobuf 3)	<pre>message PtxV2xPathDefinition { message V2xPathPoint { int32 seq = 1 [required = true]; // #[, 1-based] point within segment double Lat = 2 [required = true]; // [deg WGS-84, -90..+90, pos. = N, 6 decimals] double Lon = 3 [required = true]; // [deg WGS-84, -180..+180, pos. = E, 6 decimals] float dist = 4[required = true]; // [m, along whole point sequence, 1st point = 0, max. 2 decimals] float time = 5[required = true]; // [s, along whole point sequence, 1st point = 0, max. 1 decimal] } message V2xStopPoint { string id = 1 [required = true]; // VDV452/253 STOP(REC_ORT)::POINT_NO(ORT_NR) string name = 2[required = true]; // stop short name + "-" + point_nr (for debugging purposes) google.protobuf.DoubleValue Lat = 3; // [deg WGS-84, -90..+90, pos. = N] google.protobuf.DoubleValue Lon = 4; // [deg WGS-84, -180..+180, pos. = E] google.protobuf.FloatValue heading = 5; // [deg 0..360, 0 = true north, 90 = east, -1 = unknown] } message V2xPathSegment { int32 seq = 1 [required = true]; // #[, 1-based] segment within path repeated V2xPathPoint path_point = 2; // ordered list of points of the segment V2xStopPoint stop_point = 3; // stop point at the end of the segment where to call } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; string path_id = 2 [required = true]; // #[] repeated V2xPathSegment segment = 3; // ordered list of segments of the path } }</pre>
Retain	true (expiry: 50 hours; frequency: upon change)
QoS	1
Behaviour of IBIS	<p>The path ID shall be changed whenever the path is changed (e.g. when the vehicle is progressing further along its route and the look-ahead window is changed).</p> <p>A path will consist of 1..N segments. Each segment is typically terminated by a stop point (the destination of the vehicle). The last segment might not be terminated by a stop point. This can be the case if the destination of the vehicle is not at a stop point (e.g. in the depot or somewhere in the nature) or if the segment is so long that it makes no sense to provide a complete segment.</p> <p>The extent of the path shall be at least 1 kilometre. The last segment shall be provided fully including its stop point unless its length is greater than 10 kilometres.</p> <p>The coordinates of the stop point will in general not match the last coordinate pair of the point sequence of the path leading to the stop. Neither will the coordinates of the paths match the precise coordinates of the lane that the vehicle will use at the intersections.</p>

Attribute definitions:

Attribute	Definition
path id	ID of the path. Unique within a power cycle. Not carrying any meaning. The path is changed whenever the vehicle the path needs to be extended as per the rule defined in "behaviour of IBIS".
segment	A list of substructures. Each substructure pertains to a segment of the path. The non-last segments are terminated by a stop point at which the vehicle will call. The last segment may or not be terminated by a stop point.
segment .seq	The 1-based sequence number of the segment within the path.

Attribute	Definition
segment .path point	A list of substructures constituting a string of path points that lead to the next stop point. Each substructure pertains to such a path point. The segment point coordinates will in general correspond to the "ways" in OpenStreetMap. This means that the error will typically be in the range of a few meters.
segment .path point .seq	The 1-based sequence number of the path point within the segment.
segment .path point .lat	The latitude of the path point in degrees WGS-84, -90..+90, positive = north (6 or 7 decimals).
segment .path point .lon	The longitude of the path point in degrees WGS-84, -180..+180, positive = east (6 or 7 decimals).
segment .path point .dist	The distance in meters along the path point sequence within the whole path (not just within the segment). The first point of the first segment has distance 0.
segment .path point .time	The travel time in seconds along the path point sequence within the whole path (not just within the segment). The first point of the first segment has travel time 0. Any dwell times at the stops are not considered.
segment .stop point	A single substructure. If present, the substructure constitutes the stop point that terminates the segment. It groups the attributes of the stop point. The stop point coordinates will in general not be very precise. It is furthermore important to consider that the vehicle will in general not stop precisely at this stop point. Not only will there be a lateral difference. The longitudinal distance will typically be greater. In case of long platforms, vehicles may stop behind each other.
segment .stop point .id	The ID of the stop point at which the vehicle calls. Ref.: VDV452/253 stop(rec_ort)::point_no(ort_nr)
segment .stop point .name	The name of the stop point as a string. This name is provided for debugging purposes. It is typically constructed from the stop short name and the stop point number within the stop.
segment .stop point .lat	The latitude of the stop point in degrees WGS-84, -90..+90, positive = north (6 or 7 decimals).
segment .stop point .lon	The longitude of the stop point in degrees WGS-84, -180..+180, positive = east (6 or 7 decimals).
segment .stop point .heading	The heading of the stop point in degrees, 0..360, 90 = east (0..1 decimals). The heading will in general be quite accurate, but this is not guaranteed. In some cases, the heading has been observed to be pointing 180° into the wrong direction.

8.2.3 Path Location

This message lets the IBIS provide the current location to the OBU, so that the latter can send the SRM with the correct content at the right time.

Item	Definition
Name	PtxV2xPathLocation
Sub Topic	v2x/<v2x device>/path/location [type (always "v2x") and ID of subscribing device]
Structure (Protobuf 3)	<pre>message PtxV2xPathLocation { message V2xPathLocation { string path_id = 1 [required = true]; // reference to path int32 segment_seq = 2 [required = true]; // 1-based sequence number of segment within path int32 point_seq = 3 [required = true]; // 1-based sequence number of point within segment float dist = 4 [required = true]; // [m] distance of current location since beginning of path } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; V2xPathLocation path_loc = 2; // path location (on "before_stop" path section) }</pre>
Retain	false (frequency: every 0.5..2 seconds)
QoS	0
Behaviour of IBIS	The IBIS shall provide a path location update with a frequency of 0.2..2 Hz. If the path ID does not match the latest path that has been defined earlier, the path location shall be discarded.
Behaviour of OBU	Based on the location along the path, the OBU shall request priority for the correct lane at the upcoming intersections. If the location cannot be processed due to any inconsistencies, a log message of level "warning" shall be published to the broker.

Attribute definitions:

Attribute	Definition
path loc	A single substructure. It expresses the location of the vehicle on the current path.
path loc .path id	The reference to the path to which this location belongs. If this reference doesn't match the ID of the current path, the location shall be discarded.
path loc .segment seq	The segment of the path on which the vehicle is currently located.
path loc .point seq	The point of the segment at which the vehicle is currently located or that it has last passed.
path loc .dist	The distance of the current vehicle with respect to the beginning of the path.

8.2.4 R09 Request

This message lets the IBIS send an R09 payload to the OBU, so that the latter can request priority, presumably by embedding the payload within a CAM or SRM message.

Item	Definition
Name	PtxV2xR09Request
Sub Topic	v2x/<v2x device>/r09request/<intersection nr>/<reporting point nr> [type (always "v2x") and ID of subscribing device]
Structure (Protobuf 3)	<pre>message PtxV2xR09Request { io.ptx.proto.PtxHeader msg_header = 1 [required = true]; int32 transaction_id = 2 [required = true]; // [#, 1-based] string payload_hex = 3 [required = true]; // [HEX string] }</pre>
Retain	false
QoS	2
Behaviour of Publisher	<p>The IBIS shall publish this message according to the usual rules, when reaching a reporting point or when the driver has pushed the “priority request” button (i.e. <u>timing is important</u>).</p> <p>The purpose of providing <intersection nr> in the topic is for debugging or analysis: The messages that pertain to a single announce-request-cleardown sequence can more be easily identified or filtered.</p> <p>If <intersection nr> is unknown, an empty string can be provided.</p>

Considerations: Trapeze has implemented not only the R09.x telegrams as specified by VÖV (later VDV), but a plethora of city-specific variants thereof. To achieve migration for all of these deployments from analogue radio to V2X, sending the pre-serialised payload is the preferable option, eliminating the need for the OBU to know all of these city-specific variants.

Attribute definitions:

Attribute	Definition
transaction id	ID of the transaction. Starting at 1 upon boot of the IBIS, this number is incremented by one whenever a new reporting point is triggered.
payload_hex	The serialised R09.16 message as a hex string. When converted to binary, it is identical to the message that may be sent in parallel over the analogue radio frequency.

8.3 OBU-to-IBIS Messages

The messages in this section are sent from the OBU to the IBIS.

8.3.1 Capabilities

The OBU shall advertise its capabilities in terms of the messages and/or services it supports and their respective versions.

Item	Definition
Name	PtxV2xCapabilities
Pub Topic	v2x/<v2x device>/capabilities [type (always "v2x") and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxV2xCapabilities { message V2xServiceCapability { PtxV2xEnum.V2xServiceType type = 1 [required = true]; // supported service int32 version = 2 [required = true]; // service version (PTX, currently "1") } message V2xMessageCapability { PtxV2xEnum.V2xMessageType type = 1 [required = true]; // supported message type google.protobuf.Int32Value version = 2 [required = true]; // ETSI, currently "2" } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; repeated V2xServiceCapability service = 2; // supported services repeated V2xMessageCapability incoming_msg = 3; // supported incoming message types, can be null or empty repeated V2xMessageCapability outgoing_msg = 4; // supported outgoing message types, can be null or empty repeated PtxV2xEnum.V2xEncodingRule supported_rule = 5; // supported rules, can be null or empty }</pre>
Retain	true (expiry: 50 hours; frequency: every 24 hours or upon change)
QoS	1
Behaviour of OBU	With this message, the OBU can announce its capabilities in terms of supported services and/or messages and their versions.
Behaviour of IBIS	The IBIS shall use these capabilities message to set up the configuration message that it publishes towards the OBU after receiving the capabilities.

Attribute definitions:

Attribute	Definition
service	A list of substructures. Each substructure pertains to a service in the sense of this specification that is supported by the OBU.
service.type	The service which is supported.
service.version	The version of this specification that is supported for this service.
incoming msg	A list of substructures. Each substructure pertains to an incoming V2X message that is supported by the OBU.
incoming msg.type	The V2X message which is supported.
incoming msg.version	The version of the ETSI specification that is supported for this message.
outgoing msg	A list of substructures. Each substructure pertains to an outgoing V2X message that is supported by the OBU.

Attribute	Definition
outgoing msg .type	The V2X message which is supported.
outgoing msg .version	The version of the ETSI specification that is supported for this message.
supported rule	A list of enum values. Each value denotes a supported encoding rule for air interface mirroring. Note: Even though never used for air interface mirroring, also PCAP capability must be advertised (if supported), to indicate that the PCAP dumps will be published on request ("trigger").

8.3.2 R09 Response

If the R09 request is sent out over SRM, there could be a response from the junction controller. This message shall convey the response that has been received in an SSM to the IBIS.

Matching requests to responses is the task of the OBU.

Item	Definition
Name	PtxV2xR09Response
Pub Topic	v2x/<v2x device>/r09response/<intersection nr>/<reporting point nr> [type (always "v2x") and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxV2xR09Response { io.ptx.proto.PtxHeader msg_header = 1 [required = true]; int32 transaction_id = 2 [required = true]; PtxV2xEnum.V2xPriorityStatus priority_status = 3 [required = true]; // priority request status float distance_to_stop_line = 4 [required = true]; // [m, max. 1 decimal] }</pre>
Retain	false
QoS	2
Behaviour of OBU	The OBU shall match the SSM that corresponds to a previously sent R09-over-SRM and forward the priority status and possibly the "reason" to the IBIS. The transaction ID shall be the one that was contained in the R09 request message. If the status changes repeatedly for the same request, then there will be more than one response with the same transaction ID.
Behaviour of IBIS	The IBIS will display the priority status related to the closest intersection to the driver.

Attribute definitions:

Attribute	Definition
transaction id	The reference to the transaction to which this response belongs. If this reference doesn't match the ID of a current transaction, the response shall be discarded.
priority status	The status of the priority request as received from the traffic controller.
distance to stop line	The air distance to the stop line of the intersection that answered the R09 request and provided the status information. This value can be used by the IBIS to determine for which intersection (usually the closest) to display status information to the driver.

8.3.3 Intersection Map

This message lets the OBU provide information about a visible intersection to the IBIS. The information originates from MAP.

Item	Definition
Name	PtxV2xIntersectionMap
Pub Topic	v2x/<v2x device>/intersection/<intersection id>/map [type (always "v2x") and ID of publishing device]
Structure (Protobuf 3)	<pre> message PtxV2xIntersectionMap { message V2xGeoPoint { double lat = 1 [required = true]; // [deg WGS-84, -90..+90, pos. = N] double lon = 2 [required = true]; // [deg WGS-84, -180..+180, pos. = E] } message V2xIntersectionLane { message V2xDirectionUse { bool is_ingress = 1; bool is_egress = 2; } message V2xLaneUse { bool mixed_traffic = 1; bool nonmotor_traffic = 2; bool motor_traffic = 3; bool bus_traffic = 4; bool taxi_traffic = 5; bool pedestrian_traffic = 6; bool cyclist_traffic = 7; bool rail_traffic = 8; bool other_traffic = 9; } message V2xLaneConnection { message V2xAllowedManoeuvres { bool straight_allowed = 1; // straight movement allowed bool left_allowed = 2; // left turn allowed bool right_allowed = 3; // right turn allowed bool u_turn_allowed = 4; // u-turn allowed bool left_on_red_allowed = 5; // stop, then proceed left bool right_on_red_allowed = 6; // stop, then proceed right bool lane_change_allowed = 7; // lane change within the conflict zone bool no_stopping_allowed = 8; // should not stop at stop line bool yield_always_required = 9; // allowed movements are not protected bool go_with_halt = 10; // stop fully, then proceed bool caution = 11; // proceed with caution (conflicting traffic) } int32 signal_group_id = 1 [required = true]; // ID of movement (unique within intersection) int32 Lane_id = 2 [required = true]; // ID of the connected Lane (MUST be in same intersect.) V2xAllowedManoeuvres manoeuvres = 3; // flags for allowed manoeuvres, can be null or empty } int32 Lane_id = 1 [required = true]; // ID of the Lane within the intersection int32 approach_nr = 2 [required = true]; // approach of the intersect. to which this Lane belongs int32 Lane_nr = 3 [required = true]; // number of Lane within approach (from center to side) string name = 4 [required = true]; // for debugging only repeated V2xGeoPoint lane_point = 5 [required = true]; // from stop line outwards, should be non-empty V2xDirectionUse direction_use = 6 [required = true]; // ingress and/or egress V2xLaneUse lane_use = 7 [required = true]; // individual, buses, taxis, cyclists, pedestrians, ? repeated V2xLaneConnection connection = 8; // all egress Lanes for this ingress, can be null or empty } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; string intersection_id = 2 [required = true]; // ID of the intersection (e.g. "<region>:<id>") string name = 3 [required = true]; // for debugging only int32 revision = 4 [required = true]; // for debugging only V2xGeoPoint reference_point = 5 [required = true]; // the reference point of the intersection repeated V2xIntersectionLane Lane = 6; // set of lanes (only lanes for vehicles), should be non-empty } </pre>
Retain	true (expiry: 50 hours; frequency: upon change or change of visibility)
QoS	1

Item	Definition
Behaviour of OBU	An updated message shall be published whenever a relevant intersection has become visible (or its MAP has been updated) in its entirety (all fragments received) and has changed since the last published message. As a general principle, the OBU shall provide all attributes that are present in the MAP message. All other attributes are optional and need not be present in this message.

Attribute definitions:

Attribute	Definition
intersection id	The ID of the intersection must be unique within a reasonable geographical scope. It should be constructed as <region>:<id>.
name	The name of the intersection is for debugging only. It is presumably understandable by humans.
revision	The revision of the intersection is for debugging only. It is presumably understandable by humans.
reference point	A single substructure. The reference point is typically located at the centre of the conflict zone of the intersection. Its use for the IBIS is to determine – from all the currently visible intersections – which is the closest one and therefore currently relevant for the driver.
reference point.latitude	The latitude of the reference point in degrees WGS-84, -90..+90, positive = north (6 or 7 decimals).
reference point.longitude	The longitude of the reference point in degrees WGS-84, -180..+180, positive = east (6 or 7 decimals).
lane	A list of substructures. Each substructure pertains to a lane of the intersection.
lane.lane id	The ID of the lane within the intersection.
lane.approach nr	The intersection has fewer approaches than lanes, as each approach can have multiple lanes. The approach number identifies the approach and groups the lanes.
lane.lane nr	The number of the lane within the approach (counted from the centre to the side).
lane.name	The name of the lane is for debugging only. It is presumably understandable by humans.
lane.lane point	A list of substructures that form a string of geo points of the lane, from the stop lane outwards. This means that for ingress lanes, the string goes against the driving direction.
lane.lane point.latitude	The latitude of the lane point in degrees WGS-84, -90..+90, positive = north (6 or 7 decimals).
lane.lane point.longitude	The longitude of the lane point in degrees WGS-84, -180..+180, positive = east (6 or 7 decimals).
lane.direction use	A single substructure. The substructure tells in which directions (relative to the intersection) the lane can be used: ingress, egress, or both.
lane.direction use.is ingress	If true, the lane can be used for ingress, i.e. for driving into the intersection. Default if omitted: false.
lane.direction use.is egress	If true, the lane can be used for egress, i.e. for driving out of the intersection. Default if omitted: false.
lane.lane use	A single substructure. The substructure tells which types of vehicle may use the lane.

Attribute	Definition
lane .lane use .mixed traffic	If true, the lane can be used for mixed traffic. Default if omitted: false.
lane .lane use .nonmotor traffic	If true, the lane can be used for non-motor traffic. Default if omitted: false.
lane .lane use .motor traffic	If true, the lane can be used for motor traffic. Default if omitted: false.
lane .lane use .bus traffic	If true, the lane can be used for bus traffic. Default if omitted: false.
lane .lane use .taxi traffic	If true, the lane can be used for taxi traffic. Default if omitted: false.
lane .lane use .pedestrian traffic	If true, the lane can be used for pedestrian traffic. Default if omitted: false.
lane .lane use .cyclist traffic	If true, the lane can be used for cyclist traffic. Default if omitted: false.
lane .lane use .rail traffic	If true, the lane can be used for rail traffic. Default if omitted: false.
lane .lane use .other traffic	If true, the lane can be used for other traffic. Default if omitted: false.
lane .connection	A list of substructures. Each substructure identifies a connected egress lane and a signal group which governs the use of the conflict zone to get from this lane (in the role of ingress lane) to the connected egress lane.
lane .connection .signal group id	The ID of the signal group that is relevant to get from this lane (in the role of ingress lane) to the connected egress lane.
lane .connection .lane id	The ID of the connected egress lane.
lane .connection .manoeuvres	A single substructure. The substructure tells which manoeuvres are allowed for this lane connection. The manoeuvres are statically defined herein. They are complemented with the priority status information.
lane .manoeuvres .straight allowed	Going straight is allowed (traffic signal permitting). Default if omitted: false.

Attribute	Definition
lane .manoeuvres .left allowed	Turning left is allowed (traffic signal permitting). Default if omitted: false.
lane .manoeuvres .right allowed	Turning right is allowed (traffic signal permitting). Default if omitted: false.
lane .manoeuvres .u-turn allowed	Taking a u-turn is allowed (traffic signal permitting). Default if omitted: false.
lane .manoeuvres .left-on-red allowed	Turning left-on-red is allowed (traffic signal permitting). Default if omitted: false.
lane .manoeuvres .right-on-red allowed	Turning right-on-red is allowed (traffic signal permitting). Default if omitted: false.
lane .manoeuvres .lane change allowed	Changing the lane within the conflict zone is allowed. Default if omitted: false.
lane .manoeuvres .no stopping allowed	Stopping at the stop line is discouraged. Default if omitted: false.
lane .manoeuvres .yield always required	The allowed movements are not protected. When proceeding, conflicting traffic may have priority. Default if omitted: false.
lane .manoeuvres .go with halt	The allowed movements must be preceded by a halt at the stop line. Default if omitted: false.
lane .manoeuvres .caution	Proceed with caution. Default if omitted: false.

8.3.4 Intersection Phase

This message lets the OBU provide information about the next phases at a visible intersection. The information originates from SPAT. The purpose of this message is to provide information that can be used for driver assistance: which lane to take and how to adapt the speed to get green and save energy.

Note: The message mirrors SPAT but shall be kept stable (backward and forward compatible) should SPAT change in the future. Information pertaining to different intersections must be provided as separate messages and published to separate topics (per intersection).

Prerequisite: the current path must have been provided.

Item	Definition
Name	PtxV2xIntersectionPhase
Pub Topic	v2x/<v2x device>/intersection/<intersection id>/phase [type (always "v2x") and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxV2xIntersectionPhase { message V2xMovementState { message V2xMovementEvent { message V2xTimeChangeDetails { google.protobuf.Timestamp start_time = 1 [required = true]; // [RFC 3339, no decimals] google.protobuf.Timestamp earliest_end_time = 2; // [RFC 3339, no decimals] google.protobuf.Timestamp likely_end_time = 3; // [RFC 3339, no decimals] google.protobuf.Timestamp latest_end_time = 4; // [RFC 3339, no decimals] google.protobuf.Int32Value confidence = 5; // [0%..100%] confidence for likely end time google.protobuf.Timestamp next_time = 6; // [RFC 3339, no decimals] } PtxV2xEnum.V2xMovementPhaseState event_state = 1 [required = true]; V2xTimeChangeDetails timing = 2 [required = true]; } } int32 signal_group_id = 1 [required = true]; // ID of the signal group within the intersection string name = 2 [required = true]; // for debugging only repeated V2xMovementEvent state_time_speed = 3; // should contain current and next } io.ptx.proto.PtxHeader msg_header = 1 [required = true]; string intersection_id = 2 [required = true]; // ID of the intersection (e.g. "<region>:<id>") string name = 3 [required = true]; // for debugging only int32 revision = 4 [required = true]; // for debugging only repeated int32 enabled_lane_id = 5; // should not be empty repeated V2xMovementState state = 6; // only allowed movements as per vehicle type</pre>
Retain	false (frequency: upon change or change of visibility)
QoS	1
Behaviour of OBU	An updated message shall be published whenever a relevant intersection has become visible (or its SPAT has been updated) in its entirety (all fragments received) and has changed since the last published message and the minimum interval for this RSU has expired. As a general principle, the OBU shall provide all attributes that are present in the SPAT message. All other attributes are optional and need not be present in this message.

Attribute definitions:

Attribute	Definition
intersection id	Reference to the intersection for which the information is provided. If this reference doesn't match the ID of the intersection in an intersection map, this message shall be discarded.
name	The name of the intersection is for debugging only.
revision	The revision of the intersection phase information is for debugging only.

Attribute	Definition
enabled lane id	This list of references denotes the enabled lanes of the intersection. The other lanes are apparently disabled.
state	A list of substructures. Each substructure pertains to an allowed movement, identified by the signal group ID. It is sufficient to provide signal group information that are relevant to the vehicle type. Signal groups for pedestrians are not relevant.
state .signal group id	The ID of the signal group (= movement).
state .name	The name of the signal group (= movement) is used for debugging only.
state .state time speed	A list of substructures representing the current and future phases of the signal group. The first in the list represents the current phase.
state .state time speed .event state	The phase.
state .state time speed .timing	A single substructure that groups the timing information of this phase.
state .state time speed .timing .start time	The start time of this phase.
state .state time speed .timing .earliest end time	The earliest end time of this phase.
state .state time speed .timing .likely end time	The likely end time of this phase.
state .state time speed .timing .latest end time	The latest end time of this phase.
state .state time speed .timing .confidence	The confidence that the likely end time will materialise.

Attribute	Definition
state	The current state of the vehicle.
.state time	The next time this phase is foreseen to start.
speed	The current speed of the vehicle.
.timing	The timing information for the vehicle.
.next time	The next time the vehicle is expected to reach a certain point.

8.3.5 Intersection Status

This message lets the OBU tell which lanes to use at the intersections along the currently registered path. This allows the IBIS to filter the information contained in the intersection map (MAP) and intersection phase (SPAT) information. In addition, this message provides the status of any ongoing traffic signal priority request.

Note: This message links the registered path with MAP and SPAT. Information pertaining to different intersections must be provided as separate messages and published to separate topics (per intersection).

Prerequisite: the current path must have been provided.

Item	Definition
Name	PtxV2xIntersectionStatus
Pub Topic	v2x/<v2x device>/intersection/<intersection id>/status [type (always "v2x") and ID of publishing device]
Structure (Protobuf 3)	<pre>message PtxV2xIntersectionStatus { io.ptx.proto.PtxHeader msg_header = 1 [required = true]; PtxV2xPathLocation.V2xPathLocation path_location = 2 [(protoc.gen.jsonschema.field_options).required = true]; string intersection_id = 3 [required = true]; // reference to the intersection (e.g. "<region>:<id>") int32 signal_group_id = 4 [required = true]; // ID of movement (unique within intersection) int32 ingress_lane_id = 5 [required = true]; // ID of the recommended ingress Lane (c.f. MAP/SPAT) int32 egress_lane_id = 6 [required = true]; // ID of the recommended egress Lane (c.f. MAP/SPAT) PtxV2xEnum.V2xPriorityStatus priority_status = 7 [required = true]; // priority request status google.protobuf.Timestamp recommended_departure_from_stop = 8; // [RFC 3339, no decimals] google.protobuf.FloatValue recommended_speed = 9; // [m/s, max. 2 decimals] google.protobuf.FloatValue distance_to_stop_line = 10; // [m, max. 1 decimal] }</pre>
Retain	false (frequency: upon change or change of visibility)
QoS	1
Behaviour of OBU	<p>This message conveys information contained in MAP, SPAT and SSM. MAP and SPAT shall only be processed by the OBU once all fragments have been received.</p> <p>The OBU selects the correct movement through the intersection(s) that are visible along the currently registered path and provides this information to the IBIS, including the status of the priority request that was made (if any).</p> <p>Whenever new (or changed) information has become available that is relevant for the IBIS, this message shall be sent. It is relevant if the intersection is on the currently registered path.</p> <p>As a general principle, the OBU shall provide all attributes that are present in the MAP/SPAT/SSM messages. All other attributes are optional and need not be present in this message.</p>

Attribute definitions:

Attribute	Definition
path location	A single substructure, identical to the stand-alone message "PathLocation". It expresses the point on the current path that corresponds to the stop line of the recommended ingress lane of this intersection.
path location .path id	The reference to the path to which this intersection status message belongs. If this reference doesn't match the ID of the current path, the intersection status message shall be discarded.
path location .segment seq	The segment of the path on which the stop line of the recommended ingress lane is located.
path location .point seq	The point of the segment that precedes the stop line of the recommended ingress lane.
path location .dist	The distance of the stop line of the recommended ingress lane with respect to the beginning of the path.

Attribute	Definition
intersection id	The ID of the intersection for which this status information is provided.
signal group id	The signal group ID that is to be used to traverse the intersection.
ingress lane id	The recommended ingress lane.
egress lane id	The recommended egress lane.
priority status	The priority status that the traffic controller provided. Before receiving a response, the status is UNKNOWN. If there is no response, the status is TIMEOUT.
recommended departure from stop	The OBU may recommend when the driver should depart from the stop in order to catch the next green phase. This is an optional feature for the OBU to implement.
recommended speed	The OBU may recommend a speed which the driver should use to catch the next green phase. This is an optional feature for the OBU to implement.
distance to stop line	The OBU may provide the distance of the vehicle to the intersection. This is an optional feature for the OBU to implement.

8.4 Air Interface Mirroring

Air interface mirroring is a means

- to understand what is going on in the air at a certain location
- to find out if interesting new features could be implemented based on the available information
- to validate the correct implementation of both the own OBU and of the peer stations (OBUs, RSUs)

If mirroring is enabled for one or more incoming or outgoing message types, the messages that

- are to be sent by the OBU
- have been received by the OBU

are published by the OBU to the MQTT broker 1:1 or at least without loss of relevant information.

The OBU performs a change of serialisation:

- over the air: ASN.1-UPER (unaligned packed encoding rules)
- on the broker: ASN.1-JER (JSON encoding rules) or ASN.1-XER (XML encoding rules) or a proprietary textual representation that is easy for the field engineer to understand

Both incoming and outgoing messages are throttled: a new message is only mirrored if either (for each peer station and message type):

- the message content has changed
- the elapsed time since the last mirrored message exceeds the time interval that the IBIS provided in its configuration message

The topic structure contains

- direction (in/out)
- message type (currently "cam", "map", "spat", "srm", "ssm"; to be extended)
- encoding rule ("text", "json", "xml", "uper", "pcap")

These message types apply even if the "extended version" (e.g. SPATEM) is used over-the-air. The encoding rule determines the MQTT payload type (text or binary). At least one of the formats JSON, XML or TEXT are expected to be supported by the OBU.

8.4.1 Sent Messages

This message mirrors a message that the OBU has sent (or is about to send) over-the-air.

Item	Definition
Name	(no name, interpretation depends on message type)
Pub Topic	v2x/<v2x device>/out/<message type>/<encoding rule> [type (always "v2x") and ID of publishing device]
Retain	false
QoS	0
Behaviour of OBU	For each message the OBU sends (out) over the air <ul style="list-style-type: none"> • if mirroring has been configured for this message type • it shall publish the text version as per the configured encoding rule • to the topic corresponding to message type and encoding rule (only last part of message type enum, in lower case, e.g. "map" or "cam") (only last part of encoding rule enum, in lower-case, e.g. "json" or "text") • but not more often than the configured interval (per message type).

8.4.2 Received Messages

This message mirrors a message that the OBU has received over-the-air.

Item	Definition
Name	(no name, interpretation depends on message type)
Pub Topic	v2x/<v2x device>/in/<message type>/<encoding rule> [type (always "v2x") and ID of publishing device]
Retain	false
QoS	0
Behaviour of OBU	For each message the OBU receives (in) over the air <ul style="list-style-type: none"> • if mirroring has been configured for this message type • it shall publish the text version as per the configured encoding rule • to the topic corresponding to message type and encoding rule (only last part of message type enum, in lower case, e.g. "map" or "cam") (only last part of encoding rule enum, in lower-case, e.g. "json" or "text") • but not more often than the configured interval (per message type <u>and peer station</u>).

8.4.3 PCAP Dumps

In addition to live mirroring, the OBU might also support the creation of message dumps in PCAP format, for analysis with Wireshark. This capability needs not be announced on the broker. Access to the PCAP dumps is preferably via the OBU's built-in HTTP server.

PCAP files are kept on the OBU and can be downloaded from the OBU's HTTP server for analysis. The OBU is responsible for disk space management, i.e. to limit the disk space used for PCAP files and clean up old PCAP files.

Recommendation: Per-message logging produces lots of write cycles which could reduce the lifetime of the disk (or any non-volatile storage). It is hence recommended that the OBU write the PCAP file in larger chunks.

9 Interaction Diagrams

This chapter is INFORMATIVE. It illustrates how the V2X messages (in the air) and the PTX messages (MQTT) relate to each other. The V2X over-the-air protocol is asynchronous. Each party just broadcasts. There are no request/response handshakes. Therefore, the depicted sequences cannot be more than illustrating examples.

9.1 Sending CAM [stage 0]

Sending out the CAM is a basic function of any vehicle in the C-ITS context. The CAM consists of a low-frequency and a high-frequency component:

- The low-frequency contains static information about the vehicle (e.g. vehicle type, dimensions, ...)
- The high-frequency component changes as the vehicle moves through space (e.g. geo location, heading, speed, ...)

The following PTX messages convey the information that the OBU requires to send out CAM:

- Vehicle Info (→ low-frequency information)
- Operational Status (→ high-frequency information)

The following diagram depicts this interaction:

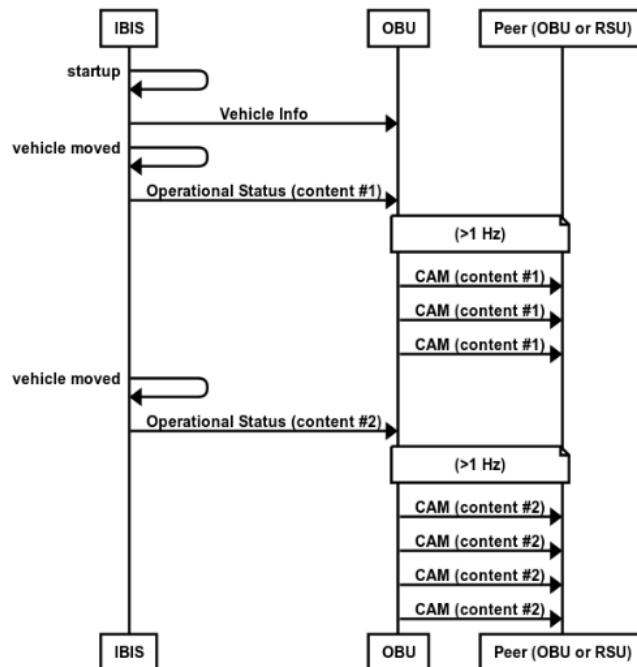


Figure 12: Interaction Diagram “Sending CAM”

Note: Whereas it is up to the OBU to complement or replace the information sent by the IBIS with information obtained from its own configuration or sensors, the integration as described in this section is highly recommended. For a fleet-wide deployment, it will reduce the overall effort and increase the data quality, especially during the maintenance and repair phase, where OBUs might be removed from one vehicle, repaired, and installed in a different vehicle.

9.2 Sending R09 over CAM [stage 1A]

The R09 “telegram” exists in as many variants as there are cities where it is used for traffic light priority over analog or digital radio. When reaching a supplied “reporting point” or triggered otherwise, at the same time as it is sent over the analog or digital channel, it shall be sent in parallel over V2X. Once all vehicles support both the “traditional” and the “new” way of sending R09 telegrams, fitting the junctions with V2X can happen at its own pace, without any further dependencies.

The OBU can wrap the R09 Request into CAM. Compared to wrapping into SRM:

- there is no response from the junction controller: the OBU cannot know whether or not the request ever reached the junction controller
- there is no hopping: the range is limited to a single “hop” and it is possible that some reporting points will have to be shifted to a different location

The following diagram depicts the interaction when wrapping the R09 request into CAM:

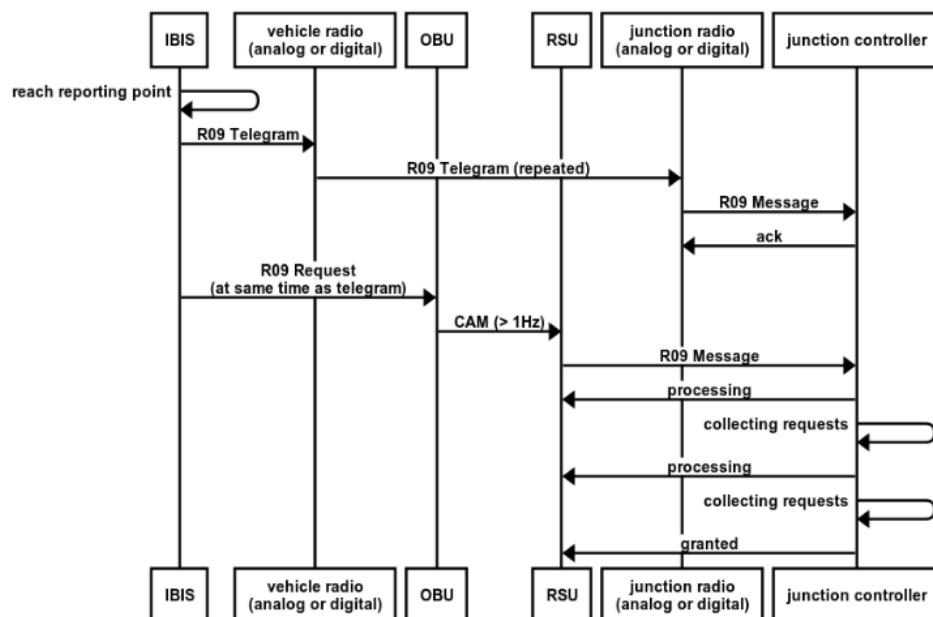


Figure 13: Interaction Diagram “Sending R09 over CAM”

As a summary, each R09 message is sent at the same time (when reaching a reporting point) over the conventional analog or digital radio and over V2X. This allows for an easy migration of the junctions.

Sending R09 telegrams over CAM is a valid replacement for analog or digital radio. It is functionally equivalent and thanks to using the robust physical layer of V2X, both the reach and the reliability of the resulting traffic light priority requests can be expected to be high.

9.3 Sending R09 over SRM [stage 1B]

The R09 “telegram” exists in as many variants as there are cities where it is used for traffic light priority over analog or digital radio. When reaching a supplied “reporting point”, at the same time as it is sent over the analog or digital channel, it shall be sent in parallel over V2X. Once all vehicles support both the “traditional” and the “new” way of sending R09 telegrams, fitting the junctions with V2X can happen at its own pace, without any further dependencies.

The OBU can wrap the R09 Request into SRM. Compared to wrapping into CAM:

- there are responses (SSM) from the junction controller: the OBU knows whether or not the request reached the junction controller
- there is hopping (between OBUs and RSUs that are “not too far”): the range is extended and it is very likely that no reporting points will have to be shifted to a different location

The following diagram depicts the interaction when wrapping the R09 request into SRM:

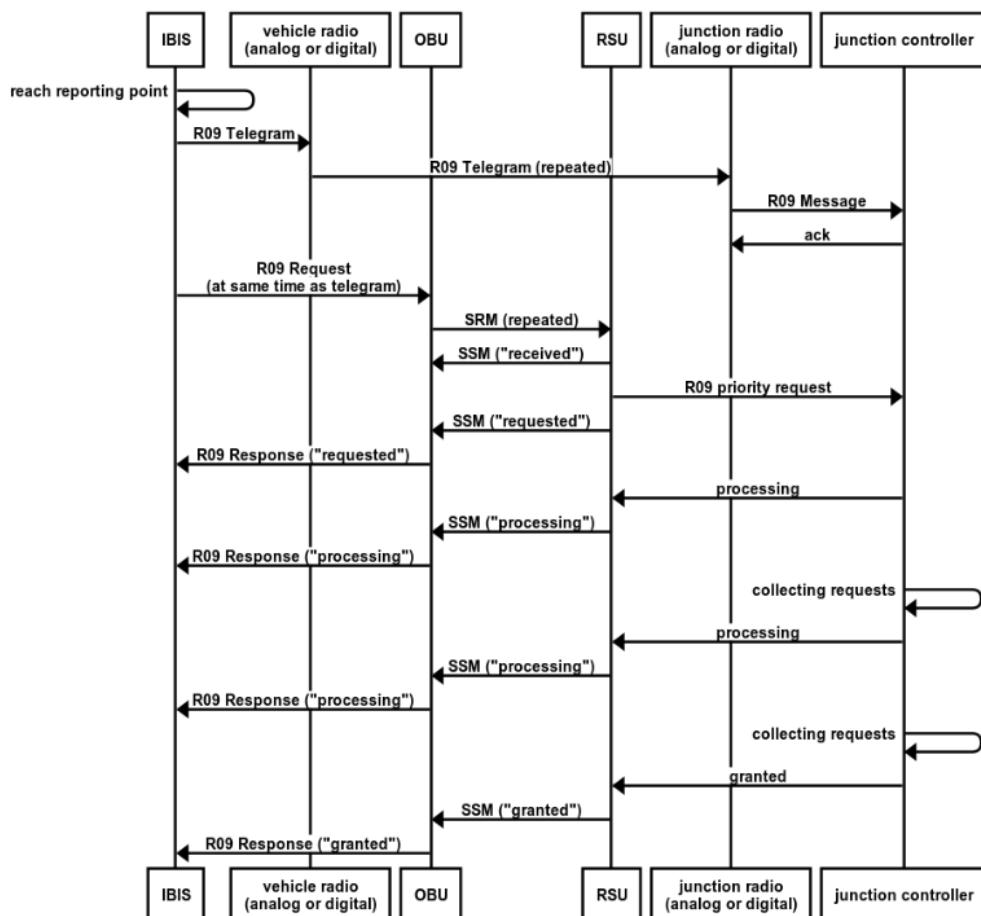


Figure 14: Interaction Diagram “Sending R09 over SRM”

To allow for an easy migration of the junctions, each R09 message is sent in parallel (when reaching a reporting point or triggered otherwise):

- over the conventional analog (VDV 420) or digital radio (VDV 426)
- over V2X

For each outgoing R09 message, multiple responses are to be expected by the IBIS, each response containing an updated status of the request.

Compared to sending R09 telegrams over CAM, sending R09 telegrams over SRM improves both range and reliability. From an IBIS perspective, the two cases differ by the R09 response which is only available when the SRM-SSM handshake is used.

9.4 Sending SRM with ETA [stage 2A]

One of the main benefits of the V2X approach to traffic light priority is that reporting points are no longer needed. The OBU can send the ETA to the RSU which relays it to the junction

controller. As the vehicle approaches the junction, it will continuously re-calculate the ETA and inform the junction controller about ETA changes. The junction controller can thereby optimise its timing of phases.

The following diagram depicts the interaction when using the SRM to send and update the ETA:

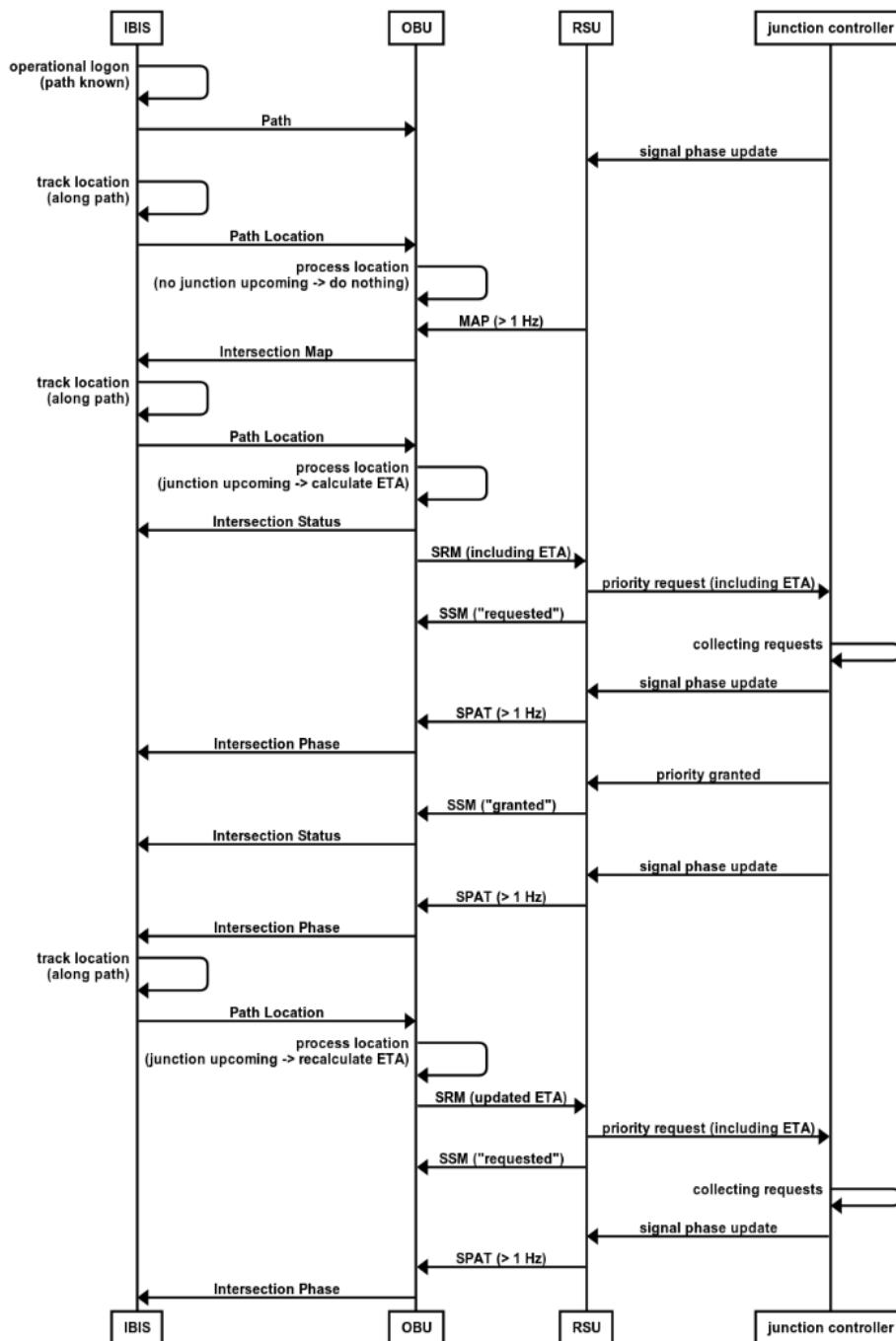


Figure 15: Interaction Diagram “Sending SRM with ETA”

As a summary, the OBU will generally propagate the V2X messages as follows:

- MAP → “Intersection Map” + “Intersection Status” (defines used lanes)
- SPAT → “Intersection Phase” + “Intersection Status” (provides phase info)
- SSM → “Intersection Status” + “Intersection Status” (provides priority status info)

Furthermore, the OBU will decide on its own when to send and update SRM, based on the information contained in "Path" and "Path Location". Reporting points are no longer needed.

9.5 Signal Phase Assistance [stage 2B]

The intended use cases of signal phase assistance are:

- energy-optimized approach to intersection ("carry momentum")
- time-optimized departure from stop ("carry passengers")

Signal phase assistance builds on the interactions of stage 2A. It does not depend on

- whether or not a priority request was made
- whether or not it was granted or rejected

All that matters is knowledge

- of the movement (ingress and egress lanes) at the next junction
- of the signal phase for this movement

This knowledge should be combined with knowledge of

- the vehicle characteristics (weight, friction, ...)
- the vertical (uphill, downhill, straight) and horizontal (curvy, straight) topography from the current vehicle location to the intersection
- any speed limits (explicit or implicit, e.g. track switches) along the path to the intersection

Appendix A – Protocol Buffer Schema Definitions

A.1 Framing (PTX_Framing_DTO.proto)

```
/**  
 * (C) 2024, Trapeze Switzerland GmbH. All rights reserved.  
 */  
  
syntax = "proto3";  
  
package io.ptx.proto; // tentative  
import "google/protobuf/timestamp.proto";  
import "chrusty/protoc-gen-jsonschema/options.proto";  
option java_multiple_files = true;  
  
message PtxHeader {  
    google.protobuf.Timestamp timestamp = 1 [(protoc.gen.jsonschema.field_options).required = true]; // [RFC 3339, 3 decimals]  
    int32 version = 2 [(protoc.gen.jsonschema.field_options).required = true]; // initial version = 1  
} // the header itself is not versioned -- only compatible extensions!!
```

A.2 Device Management (PTX_DeviceManagement_DTO.proto)

```
/**  
 * (C) 2024, Trapeze Switzerland GmbH. All rights reserved.  
 */  
  
syntax = "proto3";  
  
package io.ptx.dm.v0.proto; // tentative  
import "google/protobuf/duration.proto";  
import "google/protobuf/timestamp.proto";  
import "google/protobuf/wrappers.proto";  
  
import "chrusty/protoc-gen-jsonschema/options.proto";  
option java_multiple_files = true;  
import "PTX_Framing_DTO.proto";  
  
message PtxDmEnum {  
    enum DmDeviceModuleClassEnum {  
        CLASS_UNKNOWN = 0; // for extensibility (never used)  
        CLASS_HW = 1; // hardware  
        CLASS_FW = 2; // firmware  
        CLASS_OS = 3; // operating system  
        CLASS_SW = 4; // software  
        CLASS_DATA = 5; // data supply  
        CLASS_CFG = 6; // configuration data  
    }  
  
    enum DmDeviceReachabilityEnum { // tells whether or not status data is available at all  
        REACHABLE_UNKNOWN = 0; // for extensibility (never used)  
        REACHABLE_DIRECT = 1; // the device reports directly  
        REACHABLE_YES = 2; // the device is proxied and can be reached  
        REACHABLE_NO = 3; // the device is proxied and cannot be reached  
    }  
  
    enum DmDeviceActivationEnum { // tells whether or not status reporting makes sense  
        STATUS_UNKNOWN = 0; // used when proxied device is not reachable  
        STATUS_ACTIVE = 1; // device is active and shall be monitored  
        STATUS_INACTIVE = 2; // device is inactive (will not send any updates for a while)  
    }  
  
    enum DmDeviceHealthEnum { // general health status  
        HEALTH_UNKNOWN = 0; // used when proxied device is not reachable  
        HEALTH_OK = 1; // functionality is fully available  
        HEALTH_INFO = 2; // functionality not impaired  
        HEALTH_YELLOW = 3; // functionality impaired  
        HEALTH_RED = 4; // functionality strongly impaired or not available  
    }  
  
    enum DmDeviceTriggerEnum { // triggers the device to do something  
        TRIGGER_UNKNOWN = 0; // for extensibility (never used)  
        TRIGGER_REBOOT = 1; // instructs the device to reboot  
        TRIGGER_PUBLISH = 2; // instructs the device to publish "something"  
    }  
  
    enum DmDeviceLogLevelEnum {  
        LEVEL_UNKNOWN = 0; // for extensibility (never used)  
        LEVEL_OFF = 1; // log nothing  
        LEVEL_FATAL = 2; // fatal errors render the device useless  
        LEVEL_ERROR = 3; // errors prevent normal execution  
        LEVEL_WARNING = 4; // warnings indicate that something unexpected happened  
        LEVEL_INFO = 5; // information messages are significant to the purpose  
    }  
  
    enum DmDevicePowerStateEnum {  
        POWER_STATE_UNKNOWN = 0; // for extensibility (never used)  
        POWER_ACTIVE = 1; // "PWR" is fully available  
        SWITCH_OFF_PLANNED = 2; // "PWR" signal switched off soon (extensions possible)  
        SWITCH_OFF_IMMINENT = 3; // "PWR" signal switched off soon (no extensions)  
    }  
  
}  
  
message PtxDmPowerState {  
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];  
    PtxDmEnum.DmDevicePowerStateEnum power_state = 2 [(protoc.gen.jsonschema.field_options).required = true];  
    google.protobuf.BoolValue ignition_on = 3;  
    google.protobuf.BoolValue comm_available = 4;  
    google.protobuf.BoolValue bulk_available = 5;  
    google.protobuf.Timestamp shutdown_not_before = 6; // [RFC 3339, no decimals]  
}
```

```

message PtxDmLogLevel {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    PtxDmEnum.DmDeviceLogLevelEnum level = 2 [(protoc.gen.jsonschema.field_options).required = true];
}

message PtxDmTrigger {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    PtxDmEnum.DmDeviceTriggerEnum cmd = 2 [(protoc.gen.jsonschema.field_options).required = true];
    repeated string args = 3;
}

message PtxDmPowerRequest {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    google.protobuf.Timestamp extension = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [RFC 3339, no decimals]
    bool comm_request = 3 [(protoc.gen.jsonschema.field_options).required = true];
    bool bulk_request = 4 [(protoc.gen.jsonschema.field_options).required = true];
}

message PtxDmLogMessage {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    google.protobuf.Timestamp timestamp = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [RFC 3339, 3 decimals]
    PtxDmEnum.DmDeviceLogLevelEnum Level = 3 [(protoc.gen.jsonschema.field_options).required = true];
    string tag = 4 [(protoc.gen.jsonschema.field_options).required = true];
    string msg = 5 [(protoc.gen.jsonschema.field_options).required = true];
}

message PtxDmVersion {
    message DmModuleVersion {
        PtxDmEnum.DmDeviceModuleClassEnum class = 1 [(protoc.gen.jsonschema.field_options).required = true];
        string name = 2 [(protoc.gen.jsonschema.field_options).required = true];
        string version = 3 [(protoc.gen.jsonschema.field_options).required = true];
    }

    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    string description = 2 [(protoc.gen.jsonschema.field_options).required = true];
    repeated DmModuleVersion module = 3;
}

/***
 * This message is used in a variety of contexts:
 * - an on-board computer reports its own health
 * - an on-board computer reports the health of a connected device (OBC acts as a proxy)
 * - an on-board device reports its own health
 * - an on-board device reports the health of a connected device (the first device acts as a proxy)
 *
 * The most important attributes are:
 * - description: The description (of the device for which the health is reported) allows
 *   the device to be identified and found on-board or at the stop
 * - health: Based on this attribute, the remote monitoring personnel will be alarmed
 * - reason: Whenever the health is not "OK", the reason must be provided in a plain English
 *   sentence that is targeted to humans. If there are multiple reasons, then the reasons shall
 *   be concatenated.
 *
 * It is not sufficient to show the CPU load. If the CPU load is deemed unhealthy, the "health"
 * attribute must reflect this situation.
 */
message PtxDmHealth {
    message DmResourceUsage {
        float cpu = 1 [(protoc.gen.jsonschema.field_options).required = true]; // [0..100%, max. 1 decimal] overall CPU load (running average)
        float ram = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [0..100%, max. 1 decimal] usage of the RAM
        float disk = 3 [(protoc.gen.jsonschema.field_options).required = true]; // [0..100%, max. 1 decimal] usage of the monitored partition
    }

    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    string description = 2 [(protoc.gen.jsonschema.field_options).required = true];
    PtxDmEnum.DmDeviceReachabilityEnum reachability = 3 [(protoc.gen.jsonschema.field_options).required = true];
    PtxDmEnum.DmDeviceActivationEnum activation = 4 [(protoc.gen.jsonschema.field_options).required = true];
    PtxDmEnum.DmDeviceHealthEnum health = 5 [(protoc.gen.jsonschema.field_options).required = true];
    google.protobuf.StringValue reason = 6;
    DmResourceUsage usage = 7;
    google.protobuf.Int32Value uptime = 8; // [s] time since last boot
}

message PtxDmLogLevel {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    PtxDmEnum.DmDeviceLogLevelEnum level = 2 [(protoc.gen.jsonschema.field_options).required = true];
}

message PtxDmTrigger {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    PtxDmEnum.DmDeviceTriggerEnum cmd = 2 [(protoc.gen.jsonschema.field_options).required = true];
    repeated string args = 3;
}

message PtxDmPowerRequest {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    google.protobuf.Timestamp extension = 2 [(protoc.gen.jsonschema.field_options).required = true];
    bool comm_request = 3 [(protoc.gen.jsonschema.field_options).required = true];
    bool bulk_request = 4 [(protoc.gen.jsonschema.field_options).required = true];
}

message PtxDmLogMessage {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    google.protobuf.Timestamp timestamp = 2 [(protoc.gen.jsonschema.field_options).required = true];
    PtxDmEnum.DmDeviceLogLevelEnum Level = 3 [(protoc.gen.jsonschema.field_options).required = true];
    string tag = 4 [(protoc.gen.jsonschema.field_options).required = true];
    string msg = 5 [(protoc.gen.jsonschema.field_options).required = true];
}

message PtxDmVersion {
    message DmModuleVersion {
        PtxDmEnum.DmDeviceModuleClassEnum class = 1 [(protoc.gen.jsonschema.field_options).required = true];
        string name = 2 [(protoc.gen.jsonschema.field_options).required = true];
        string version = 3 [(protoc.gen.jsonschema.field_options).required = true];
    }

    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    string description = 2 [(protoc.gen.jsonschema.field_options).required = true];
    repeated DmModuleVersion module = 3;
}

```

```

}

/*
 * This message is used in a variety of contexts:
 * - an on-board computer reports its own health
 * - an on-board computer reports the health of a connected device (OBC acts as a proxy)
 * - an on-board device reports its own health
 * - an on-board device reports the health of a connected device (the first device acts as a proxy)
 *
 * The most important attributes are:
 * - description: The description (of the device for which the health is reported) allows
 *   the device to be identified and found on-board or at the stop
 * - health: Based on this attribute, the remote monitoring personnel will be alarmed
 * - reason: Whenever the health is not "ok", the reason must be provided in a plain English
 *   sentence that is targeted to humans. If there are multiple reasons, then the reasons shall
 *   be concatenated.
 *
 * It is not sufficient to show the CPU load. If the CPU load is deemed unhealthy, the "health"
 * attribute must reflect this situation.
 */
message PtxOmHealth {
    message DmResourceUsage {
        message DmCpuUsage {
            google.protobuf.Int32Value core = 1; // [#]
            google.protobuf.FloatValue frequency = 2; // [GHz, max. 3 decimals]
            float load = 3 [(protoc.gen.jsonschema.field_options).required = true]; // [0%..100%, max. 1 decimal]
        }

        message DmRamUsage {
            float capacity = 1 [(protoc.gen.jsonschema.field_options).required = true]; // [GByte, max. 3 decimals]
            float used = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [0%..100%, max. 1 decimal]
        }

        message DmDiskUsage {
            google.protobuf.StringValue partition = 1; // [name]
            float capacity = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [GByte, max. 3 decimals]
            float used = 3 [(protoc.gen.jsonschema.field_options).required = true]; // [0%..100%, max. 1 decimal]
        }

        repeated DmCpuUsage cpu = 1;
        DmRamUsage ram = 2;
        repeated DmDiskUsage disk = 3;
    }

    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    string description = 2 [(protoc.gen.jsonschema.field_options).required = true];
    PtxOmEnum.OmDeviceReachabilityEnum reachability = 3 [(protoc.gen.jsonschema.field_options).required = true];
    PtxOmEnum.OmDeviceActivationEnum activation = 4 [(protoc.gen.jsonschema.field_options).required = true];
    PtxOmEnum.OmDeviceHealthEnum health = 5 [(protoc.gen.jsonschema.field_options).required = true];
    google.protobuf.StringValue reason = 6;
    DmResourceUsage usage = 7;
}


```

A.3 Operational Information (PTX_OperationalInfo.DTO.proto)

```

/*
 * (C) 2024, Trapeze Switzerland GmbH. All rights reserved.
 */

syntax = "proto3";

package io.ptx.oi.v0.proto; // tentative

import "google/protobuf/duration.proto";
import "google/protobuf/timestamp.proto";
import "google/protobuf/wrappers.proto";

import "chrusty/protoc-gen-jsonschema/options.proto";

option java_multiple_files = true;

import "PTX_Framing.DTO.proto";

message PtxOiEnum {

    enum OiVehicleCategory {
        CAT_OTHER = 0; // unknown or "other"
        CAT_BUS = 1; // incl. articulated, google.protobuf.DoubleValue decker, rapid, ...
        CAT_TROLLEY = 2; // trolley bus
        CAT_TRAM = 3; // rail vehicle that may circulate on public roads
        CAT_RAIL = 4; // light rail, subway, heavy rail, magnetic rail, ...
        CAT_FUNI = 5; // rail shuttle (linear track)
        CAT_GONDOLA = 6; // suspended shuttle (linear track)
        CAT_FERRY = 7; // waterway vehicle
    }

    enum OiLocationStatus {
        LOC_UNKNOWN = 0; // for extensibility (never used)
        LOC_NONE = 1; // not currently logged onto a journey or pattern
        LOC_OFF_COURSE = 2; // logged onto a journey or pattern, off-course
        LOC_ON_COURSE = 3; // logged onto a journey or pattern, on-course
    }

    enum OiPriorityLevel {
        PRIO_UNKNOWN = 0; // for extensibility (never used)
        PRIO_NORMAL = 1; // priority to be requested
        PRIO_OFF_AT_STOP = 2; // priority not to be requested (at stop)
        PRIO_OFF = 3; // priority not to be requested (non-commercial operation)
    }

    enum OidriverCabActivation {
        CAB_UNKNOWN = 0; // for extensibility (never used)
        CAB_NONE = 1; // no cab is active
        CAB_A = 2; // cab A is active
        CAB_B = 3; // cab B is active
    }
}

message PtxOivehicleInfo {


```

```

io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
PtxoEnum.oVehicleCategory category = 2 [(protoc.gen.jsonschema.field_options).required = true];
google.protobuf.StringValue type = 3; // for debugging only, non-null and non-empty if provided
google.protobuf.BoolValue has_trailer = 4;
google.protobuf.Int32Value nof_vehicles = 5; // >1 indicates coupling (tram or train)
google.protobuf.FloatValue weight = 6; // [kg, 0 decimals] sum for whole train
google.protobuf.FloatValue length = 7; // [m, max. 2 decimals] sum for whole train
google.protobuf.FloatValue width = 8; // [m, max. 2 decimals] max for whole train
google.protobuf.FloatValue height = 9; // [m, max. 2 decimals] max for whole train
google.protobuf.StringValue plate = 11; // non-null and non-empty if provided
google.protobuf.StringValue vin = 12; // non-null and non-empty if provided
}

message PtxoOperationalLogon { // IDs refer to VDV452 and VDV4558
io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
string vehicle_id = 2 [(protoc.gen.jsonschema.field_options).required = true]; // VDV452/443 VEHICLE(FAHRZEUG)::VEHICLE_NO(FZG_NR)
google.protobuf.StringValue driver_id = 3; // VDV4558 DRIVER(PERSONAL)::DRIVER_NR(FAHRER_NR)
google.protobuf.StringValue block_id = 4; // VDV452/310 BLOCK::BLOCK_NO(UM_UID)
google.protobuf.StringValue journey_id = 5; // VDV452/715 JOURNEY(REC_FRT)::JOURNEY_NO(FRT_FID)
google.protobuf.StringValue pattern_id = 6; // VDV452/226 LINE(REC_LID)::ROUTE_NO(ROUTEN_NR)
google.protobuf.StringValue line_id = 7; // VDV452/226 LINE(REC_LID)::LINE_NO(LI_NR)
google.protobuf.StringValue direction_id = 8; // VDV452/226 LINE(REC_LID)::DIRECTION(LI_RI_NR)
google.protobuf.StringValue run_id = 9; // VDV452/715 JOURNEY(REC_FRT)::RUN(LI_KU_NR)
}

message PtxoOperationalJourney { // IDs refer to VDV452
message OiJourneyCall {
    message OiStopPoint {
        string id = 1 [(protoc.gen.jsonschema.field_options).required = true]; // if planned stop point: VDV452/253
STOP(REC_ORT)::POINT_NO(ORT_NR)
        string name = 2 [(protoc.gen.jsonschema.field_options).required = true]; // stop short name + "-" + point nr (for debugging purposes)
        google.protobuf.DoubleValue lat = 3; // [deg WGS-84, -90..+90, pos. = N] mandatory if available
        google.protobuf.DoubleValue lon = 4; // [deg WGS-84, -180..+180, pos. = E] mandatory if available
        google.protobuf.FloatValue heading = 5; // [deg 0..360, 0 = true north, 90 = east] mandatory if available
    }
    message OiCallData {
        google.protobuf.Timestamp timestamp = 1 [(protoc.gen.jsonschema.field_options).required = true]; // [RFC 3339, no decimals]
    }

    int32 call_seq = 1 [(protoc.gen.jsonschema.field_options).required = true]; // #[, 1-based]
    OiStopPoint stop_point = 2 [(protoc.gen.jsonschema.field_options).required = true]; // stop point at which the vehicle calls
    OiCallData arrival_data = 3; // information pertaining to the arrival
    OiCallData departure_data = 4; // information pertaining to the departure
    google.protobuf.FloatValue dist_to_next_stop = 5; // [m, max. 1 decimal]
    google.protobuf.BoolValue do_not_dwell = 6; // indication that vehicle must continue even if ahead of time
    google.protobuf.Int32Value typical_dwell_time = 7; // [s]
}
}

io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
google.protobuf.StringValue journey_id = 2 [(protoc.gen.jsonschema.field_options).required = true]; // VDV452/715
JOURNEY(REC_FRT)::JOURNEY_NO(FRT_FID)
repeated OiJourneyCall call = 3; // ordered list of scheduled calls of this journey
}

message PtxoOperationalStatus {
    message OiVehicleSignals {
        google.protobuf.BoolValue reverse_gear = 1;
        google.protobuf.BoolValue doors_released = 2;
        google.protobuf.BoolValue doors_open = 3;
        google.protobuf.BoolValue stop_brake_active = 4;
    }

    message OiGeoLocation {
        double latitude = 1 [(protoc.gen.jsonschema.field_options).required = true]; // [deg WGS-84, -90..+90, pos. = N, 6 decimals]
        double longitude = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [deg WGS-84, -180..+180, pos. = E, 6 decimals]
        google.protobuf.FloatValue accuracy = 3; // [m, non-neg., max. 1 decimals]
        google.protobuf.FloatValue altitude = 4; // [m, pos. = up, 1 decimal]
        google.protobuf.FloatValue vertical_accuracy = 5; // [m, non-neg., max. 1 decimal]
        google.protobuf.FloatValue heading = 6; // [deg 0..360, 0 = true north, 90 = east, max. 1 decimal]
        google.protobuf.FloatValue speed = 7; // [m/s, max. 2 decimals] speed from GNSS
    }

    message OiLogicalLocation {
        int32 call_seq = 1 [(protoc.gen.jsonschema.field_options).required = true]; // #[, 1-based] prev-or-current-or-next stop
        google.protobuf.FloatValue distance = 2; // [m, max. 1 decimal] neg. = before stop, pos. = after stop
    }

    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    PtxoEnum.oDriverCabActivation driver_cab_active = 2 [(protoc.gen.jsonschema.field_options).required = true];
    OiVehicleSignals vehicle_signals = 3;
    google.protobuf.FloatValue odo_speed = 4; // [m/s]
    google.protobuf.Int32Value sat_count = 5; // #[#
    OiGeoLocation geo_loc = 6;
    PtxoEnum.oLocationStatus status = 7 [(protoc.gen.jsonschema.field_options).required = true];
    OiLogicalLocation logical_loc = 8; // only if logged onto a journey
    google.protobuf.Int32Value deviation = 10; // [s, pos. = delayed]
    google.protobuf.FloatValue occupancy = 11; // [%..100, max. 1 decimal]
    PtxoEnum.oPriorityLevel prio_level = 12 [(protoc.gen.jsonschema.field_options).required = true];
}
}

```

A.4 Vehicle to Everything (PTX_VehicleToEverything_DTO.proto)

```

/**
 * (C) 2024, Trapeze Switzerland GmbH. All rights reserved.
 */
syntax = "proto3";
package io.ptx.v2x.v0.proto; // tentative
import "google/protobuf/timestamp.proto";
import "google/protobuf/wrappers.proto";
import "crusty/protoc-gen-jsonschema/options.proto";
option java_multiple_files = true;

```

```

import "PTX_Framing.DTO.proto";

message PtxV2xEnum {

enum V2xEncodingRule {
    ENCODING_UNKNOWN = 0; // for extensibility (can be used to disable mirroring)
    ENCODING_TEXT = 1; // mirrored messages converted to human-readable text
    ENCODING_UPER = 2; // the unaligned packed encoding used over-the-air (ASN.1-UPER)
    ENCODING_JSON = 3; // the canonical JSON text format (ASN.1-JER)
    ENCODING_XML = 4; // the canonical XML text format (ASN.1-XER)
    ENCODING_PCAP = 5; // PCAP (e.g. for viewing with wireshark)
}

enum V2xServiceType {
    SERVICE_UNKNOWN = 0; // for extensibility (never used)
    SERVICE_R09_REQUEST_ONLY = 1; // sends R09 payload embedded in CAM
    SERVICE_R09_REQUEST_RESPONSE = 2; // sends R09 payload embedded in SRM, receives SSM
    SERVICE_PHASE = 3; // understands path, receives MAP and SPAT
    SERVICE_PRIORITY = 4; // understands path, receives MAP, sends SRM, receives SSM
    SERVICE_MAKE_AWARE = 5; // sends CAM
}

enum V2xMessageType {
    MESSAGE_UNKNOWN = 0; // for extensibility (never used)
    MESSAGE_CAM = 1; // co-operative awareness
    MESSAGE_MAP = 2; // intersection map
    MESSAGE_SPAT = 3; // signal phase and timing
    MESSAGE_SRM = 4; // signal request
    MESSAGE_SSM = 5; // signal status
}

enum V2xMovementPhaseState {
    PHASE_UNAVAILABLE = 0; // SPAT::MovementPhaseState::unavailable (0)
    PHASE_DARK = 1; // SPAT::MovementPhaseState::dark (1)
    PHASE_FLASHING_RED = 2; // SPAT::MovementPhaseState::stop-then-proceed (2)
    PHASE_RED = 3; // SPAT::MovementPhaseState::stop-and-remain (3)
    PHASE_RED_AND_YELLOW = 4; // SPAT::MovementPhaseState::pre-movement (4)
    PHASE_GREEN = 5; // SPAT::MovementPhaseState::permissive-movement-allowed (5)
    PHASE_GREEN_EXCLUSIVE = 6; // SPAT::MovementPhaseState::protected-movement-allowed (6)
    PHASE_YELLOW = 7; // SPAT::MovementPhaseState::permissive-clearance (7)
    PHASE_YELLOW_EXCLUSIVE = 8; // SPAT::MovementPhaseState::protected-clearance (8)
    PHASE_FLASHING_YELLOW = 9; // SPAT::MovementPhaseState::caution-conflicting-traffic (9)
}

enum V2xPriorityStatus {
    STATUS_UNKNOWN = 0; // SSM::PrioritizationResponseStatus::unknown(0)
    STATUS_REQUESTED = 1; // SSM::PrioritizationResponseStatus::requested(1)
    STATUS_PROCESSING = 2; // SSM::PrioritizationResponseStatus::processing(2)
    STATUS_TRAFFIC = 3; // SSM::PrioritizationResponseStatus::watchOtherTraffic(3)
    STATUS_GRANTED = 4; // SSM::PrioritizationResponseStatus::granted(4)
    STATUS_REJECTED = 5; // SSM::PrioritizationResponseStatus::rejected(5)
    STATUS_MAX = 6; // SSM::PrioritizationResponseStatus::maxPresence(6)
    STATUS_LOCKED = 7; // SSM::PrioritizationResponseStatus::serviceLocked(7)
    STATUS_TIMEOUT = 8; // priority request has timed out (no answer)
}
}

message PtxV2xConfiguration {

message V2xServiceConfig {
    PtxV2xEnum.V2xServiceType type = 1 [(protoc.gen.jsonschema.field_options).required = true]; // service to be activated
    int32 interval = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [s] min. interval (per station) for OBU-to-IBIS messages
}

message V2xMessageConfig {
    PtxV2xEnum.V2xMessageType type = 1 [(protoc.gen.jsonschema.field_options).required = true]; // mirror messages of this type
    int32 interval = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [s] min. interval per message of this type
}

io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
repeated V2xServiceConfig service = 2; // services to be activated, can be null or empty
repeated V2xMessageConfig incoming_msg = 3; // mirrored incoming messages, can be null or empty
repeated V2xMessageConfig outgoing_msg = 4; // mirrored outgoing messages, can be null or empty
PtxV2xEnum.V2xEncodingRule selected_rule = 5; // encoding rule for mirroring
}

message PtxV2xpathDefinition {

message V2xPathPoint {
    int32 seq = 1 [(protoc.gen.jsonschema.field_options).required = true]; // #[, 1-based] point within segment
    double lat = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [deg WGS-84, -90..+90, pos. = N, 6 decimals]
    double lon = 3 [(protoc.gen.jsonschema.field_options).required = true]; // [deg WGS-84, -180..+180, pos. = E, 6 decimals]
    float dist = 4 [(protoc.gen.jsonschema.field_options).required = true]; // [m, along whole point sequence, 1st point = 0, max. 2 decimals]
    float time = 5 [(protoc.gen.jsonschema.field_options).required = true]; // [s, along whole point sequence, 1st point = 0, max. 1 decimal]
}

message V2xStopPoint {
    string id = 1 [(protoc.gen.jsonschema.field_options).required = true]; // VDV452/253 STOP(REC_ORT)::POINT_NO(ORT_NR)
    string name = 2 [(protoc.gen.jsonschema.field_options).required = true]; // stop short name + "-" + point nr (for debugging purposes)
    google.protobuf.DoubleValue lat = 3; // [deg WGS-84, -90..+90, pos. = N]
    google.protobuf.DoubleValue lon = 4; // [deg WGS-84, -180..+180, pos. = E]
    google.protobuf.FloatValue heading = 5; // [deg 0..360, 0 = true north, 90 = east, -1 = unknown]
}

message V2xPathSegment {
    int32 seq = 1 [(protoc.gen.jsonschema.field_options).required = true]; // #[, 1-based] segment within path
    repeated V2xPathPoint path_point = 2; // ordered list of points of the segment
    V2xStopPoint stop_point = 3; // stop point at the end of the segment where to call
}

io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
string path_id = 2 [(protoc.gen.jsonschema.field_options).required = true]; // #[]
repeated V2xPathSegment segment = 3; // ordered list of segments of the path
}

message PtxV2xpathLocation {

message V2xPathLocation {
    string path_id = 1 [(protoc.gen.jsonschema.field_options).required = true]; // reference to path
    int32 segment_seq = 2 [(protoc.gen.jsonschema.field_options).required = true]; // 1-based sequence number of segment within path
    int32 point_seq = 3 [(protoc.gen.jsonschema.field_options).required = true]; // 1-based sequence number of point within segment
    float dist = 4 [(protoc.gen.jsonschema.field_options).required = true]; // [m] distance of current location since beginning of path
}

io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
V2xPathLocation path_loc = 2; // path location (on "before_stop" path section)
}

```

```

}

message PtxV2xR09Request {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    int32 transaction_id = 2 [(protoc.gen.jsonschema.field_options).required = true]; // #[, 1-based]
    string payload_hex = 3 [(protoc.gen.jsonschema.field_options).required = true]; // [HEX string]
}

message PtxV2xCapabilities {
    message V2xServiceCapability {
        PtxV2xEnum.V2xServiceType type = 1 [(protoc.gen.jsonschema.field_options).required = true]; // supported service
        int32 version = 2 [(protoc.gen.jsonschema.field_options).required = true]; // service version (PTX, currently "1")
    }

    message V2xMessageCapability {
        PtxV2xEnum.V2xMessageType type = 1 [(protoc.gen.jsonschema.field_options).required = true]; // supported message type
        google.protobuf.Int32Value version = 2 [(protoc.gen.jsonschema.field_options).required = true]; // ETSI, currently "2"
    }

    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    repeated V2xServiceCapability service = 2; // supported services
    repeated V2xMessageCapability incoming_msg = 3; // supported incoming message types, can be null or empty
    repeated V2xMessageCapability outgoing_msg = 4; // supported outgoing message types, can be null or empty
    repeated PtxV2xEnum.V2xEncodingRule supported_rule = 5; // supported rules, can be null or empty
}

message PtxV2xR09Response {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    int32 transaction_id = 2 [(protoc.gen.jsonschema.field_options).required = true];
    PtxV2xEnum.V2xPriorityStatus priority_status = 3 [(protoc.gen.jsonschema.field_options).required = true]; // priority request status
    float distance_to_stop_line = 4 [(protoc.gen.jsonschema.field_options).required = true]; // [m, max. 1 decimal]
}

message PtxV2xIntersectionMap {
    message V2xGeoPoint {
        double lat = 1 [(protoc.gen.jsonschema.field_options).required = true]; // [deg WGS-84, -90..+90, pos. = N]
        double lon = 2 [(protoc.gen.jsonschema.field_options).required = true]; // [deg WGS-84, -180..+180, pos. = E]
    }

    message V2xIntersectionLane {
        message V2xDirectionUse {
            bool is_ingress = 1 [(protoc.gen.jsonschema.field_options).required = true];
            bool is_egress = 2 [(protoc.gen.jsonschema.field_options).required = true];
        }

        message V2xLaneUse {
            bool mixed_traffic = 1 [(protoc.gen.jsonschema.field_options).required = true];
            bool nonmotor_traffic = 2 [(protoc.gen.jsonschema.field_options).required = true];
            bool motor_traffic = 3 [(protoc.gen.jsonschema.field_options).required = true];
            bool bus_traffic = 4 [(protoc.gen.jsonschema.field_options).required = true];
            bool taxi_traffic = 5 [(protoc.gen.jsonschema.field_options).required = true];
            bool pedestrian_traffic = 6 [(protoc.gen.jsonschema.field_options).required = true];
            bool cyclist_traffic = 7 [(protoc.gen.jsonschema.field_options).required = true];
            bool rail_traffic = 8 [(protoc.gen.jsonschema.field_options).required = true];
            bool other_traffic = 9 [(protoc.gen.jsonschema.field_options).required = true];
        }
    }

    message V2xLaneConnection {
        message V2xAllowedManoeuvres {
            bool straight_allowed = 1 [(protoc.gen.jsonschema.field_options).required = true]; // straight movement allowed
            bool left_allowed = 2 [(protoc.gen.jsonschema.field_options).required = true]; // left turn allowed
            bool right_allowed = 3 [(protoc.gen.jsonschema.field_options).required = true]; // right turn allowed
            bool u_turn_allowed = 4 [(protoc.gen.jsonschema.field_options).required = true]; // u-turn allowed
            bool left_on_red_allowed = 5 [(protoc.gen.jsonschema.field_options).required = true]; // stop, then proceed left
            bool right_on_red_allowed = 6 [(protoc.gen.jsonschema.field_options).required = true]; // stop, then proceed right
            bool lane_change_allowed = 7 [(protoc.gen.jsonschema.field_options).required = true]; // lane change within the conflict zone
            bool no_stopping_allowed = 8 [(protoc.gen.jsonschema.field_options).required = true]; // should not stop at stop line
            bool yield_always_required = 9 [(protoc.gen.jsonschema.field_options).required = true]; // allowed movements are not protected
            bool go_with_halt = 10 [(protoc.gen.jsonschema.field_options).required = true]; // stop fully, then proceed
            bool caution = 11 [(protoc.gen.jsonschema.field_options).required = true]; // proceed with caution (conflicting traffic)
        }

        int32 signal_group_id = 1 [(protoc.gen.jsonschema.field_options).required = true]; // ID of movement (unique within intersection)
        int32 lane_id = 2 [(protoc.gen.jsonschema.field_options).required = true]; // ID of the connected lane (MUST be in same intersect.)
        V2xAllowedManoeuvres manoeuvres = 3; // flags for allowed manoeuvres, can be null or empty
    }

    int32 lane_id = 1 [(protoc.gen.jsonschema.field_options).required = true]; // ID of the lane within the intersection
    int32 approach_nr = 2 [(protoc.gen.jsonschema.field_options).required = true]; // approach of the intersect. to which this lane belongs
    int32 lane_nr = 3 [(protoc.gen.jsonschema.field_options).required = true]; // number of lane within approach (from center to side)
    string name = 4 [(protoc.gen.jsonschema.field_options).required = true]; // for debugging only
    repeated V2xGeoPoint lane_point = 5 [(protoc.gen.jsonschema.field_options).required = true]; // from stop line outwards, should be non-empty
    V2xDirectionUse direction_use = 6 [(protoc.gen.jsonschema.field_options).required = true]; // ingress and/or egress
    V2xLaneUse lane_use = 7 [(protoc.gen.jsonschema.field_options).required = true]; // individual, buses, taxis, cyclists, pedestrians, ?
    repeated V2xLaneConnection connection = 8; // all egress lanes for this ingress, can be null or empty
}

io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
string intersection_id = 2 [(protoc.gen.jsonschema.field_options).required = true]; // ID of the intersection (e.g. "<region>:<id>")
string name = 3 [(protoc.gen.jsonschema.field_options).required = true]; // for debugging only
int32 revision = 4 [(protoc.gen.jsonschema.field_options).required = true]; // for debugging only
V2xGeoPoint reference_point = 5 [(protoc.gen.jsonschema.field_options).required = true]; // the reference point of the intersection
repeated V2xIntersectionLane lane = 6; // set of lanes (only lanes for vehicles), should be non-empty
}

message PtxV2xIntersectionPhase {
    message V2xMovementState {
        message V2xMovementEvent {
            message V2xTimeChangeDetails {
                google.protobuf.Timestamp start_time = 1 [(protoc.gen.jsonschema.field_options).required = true]; // [RFC 3339, no decimals]
                google.protobuf.Timestamp earliest_end_time = 2; // [RFC 3339, no decimals]
                google.protobuf.Timestamp likely_end_time = 3; // [RFC 3339, no decimals]
                google.protobuf.Timestamp latest_end_time = 4; // [RFC 3339, no decimals]
                google.protobuf.Int32Value confidence = 5; // [0%..100%] confidence for likely end time
                google.protobuf.Timestamp next_time = 6; // [RFC 3339, no decimals]
            }

            PtxV2xEnum.V2xMovementPhaseState event_state = 1 [(protoc.gen.jsonschema.field_options).required = true];
            V2xTimeChangeDetails timing = 2 [(protoc.gen.jsonschema.field_options).required = true];
        }
    }
}

```

```

int32 signal_group_id = 1 [(protoc.gen.jsonschema.field_options).required = true]; // ID of the signal group within the intersection
string name = 2 [(protoc.gen.jsonschema.field_options).required = true]; // for debugging only
repeated V2xMovementEvent state_time_speed = 3; // should contain current and next
}

io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
string intersection_id = 2 [(protoc.gen.jsonschema.field_options).required = true]; // ID of the intersection (e.g. "<region>:<id>")
string name = 3 [(protoc.gen.jsonschema.field_options).required = true]; // for debugging only
int32 revision = 4 [(protoc.gen.jsonschema.field_options).required = true]; // for debugging only
repeated int32 enabled_lane_id = 5; // should not be empty
repeated V2xMovementState state = 6; // only allowed movements as per vehicle type
}

message PtxV2xIntersectionstatus {
    io.ptx.proto.PtxHeader msg_header = 1 [(protoc.gen.jsonschema.field_options).required = true];
    string path_id = 2 [(protoc.gen.jsonschema.field_options).required = true]; // the path (cf. path definition) to which this message refers
    string intersection_id = 3 [(protoc.gen.jsonschema.field_options).required = true]; // reference to the intersection (e.g. "<region>:<id>")
    int32 signal_group_id = 4 [(protoc.gen.jsonschema.field_options).required = true]; // ID of movement (unique within intersection)
    int32 ingress_lane_id = 5 [(protoc.gen.jsonschema.field_options).required = true]; // ID of the recommended ingress lane (cf. MAP/SPAT)
    int32 egress_lane_id = 6 [(protoc.gen.jsonschema.field_options).required = true]; // ID of the recommended egress lane (cf. MAP/SPAT)
    PtxV2xEnum.V2xPrioritystatus priority_status = 7 [(protoc.gen.jsonschema.field_options).required = true]; // priority request status
    google.protobuf.Timestamp recommended_departure_from_stop = 8; // [RFC 3339, no decimals]
    google.protobuf.FloatValue recommended_speed = 9; // [m/s, max. 2 decimals]
    google.protobuf.FloatValue distance_to_stop_line = 10; // [m, max. 1 decimal]
}

```

Appendix B – Schema Files

The embedded JSON schemas have been generated from the Protobuf schemas by means of github.com/chrusty/protoc-gen-jsonschema by means of the following command:

```
protoc \
  --jsonschema_out=json \
  --proto_path=proto \
  --jsonschema_opt=enums_as_strings_only \
  --jsonschema_opt=enums_trim_prefix \
  --jsonschema_opt=type_names_with_no_package \
proto/*.proto
```

Please address the named contact persons (cf. Document Management) in order to obtain the current schema files.

Appendix C – Illustrations

C.1 Path Definition

This screenshot depicts a complex path setup (= reference for the explanations that follow).



Figure 16: Complex Path Setup

The vehicle (blue-green symbol approaching from the north) is logged onto a pattern that leads from the west (MICA, not entirely visible) to the east (GUET and beyond).

The pattern has been modified by a dispatch action: between FARB and HERD, the sequence FARB-SEID-BALT-LETB-LETP-SBBW-HERD (grey) has been replaced by the sequence FARB-BACH-LIND-GRIM-FREI-LETG-HERD (pink, constitutes a detour).

The vehicle has been instructed to use an insertion point (FREI), i.e. to start its journey from there. The vehicle is to follow a defined path (green, some brownish colour where it overlaps the pink detour) from its current location to the insertion point (FREI).

This scenario is reflected in the Path Definition message as follows:

1. When the vehicle logs on and as soon as the insertion point is defined, a first Path Definition message is published, containing the shape from the current vehicle location to the insertion point (green section).
2. Whenever the vehicle leaves the green path, the shape from the vehicle location to the insertion point is re-calculated and the Path Definition message is re-published (new ID).
3. As soon as the the vehicle goes on-course on the detour (pink section), a second Path Definition message is published, containing the shape of the modified pattern (pink and blue sections, possibly even extending into the return pattern). Note that in the illustrated case, the vehicle will go on-course just after passing GRIM, i.e. a few hundred meters before arriving at the insertion point.
4. Whenever the vehicle leaves the shape of the modified pattern, or when the driver manually defines the next stop to be served within the modified pattern, a green path is calculated to the next stop to be served (i.e. back to 1, with a different insertion point).

Appendix D – Sources for Interaction Diagrams

Use WebSequenceDiagrams.com/app (“patent” style) to recreate the interaction diagrams.

D.1 Sending CAM

```
participant IBIS as IBIS
participant OBU as OBU
participant Peer (OBU or RSU) as CITS

IBIS->IBIS: startup
IBIS->OBU: Vehicle Info

IBIS->IBIS: vehicle moved
IBIS->OBU: Operational Status (content #1)

note over OBU,CITS: (>1 Hz)
OBU->CITS: CAM (content #1)
OBU->CITS: CAM (content #1)
OBU->CITS: CAM (content #1)

IBIS->IBIS: vehicle moved
IBIS->OBU: Operational Status (content #2)

note over OBU,CITS: (>1 Hz)

OBU->CITS: CAM (content #2)
OBU->CITS: CAM (content #2)
OBU->CITS: CAM (content #2)
OBU->CITS: CAM (content #2)
```

D.2 Sending R09 over CAM

```
participant IBIS as IBIS
participant vehicle radio\n(analog or digital) as VRADIO
participant OBU as OBU
participant RSU as RSU
participant junction radio\n(analog or digital) as JRADIO
participant junction controller as JC

IBIS->IBIS: reach reporting point
IBIS->VRADIO: R09 Telegram
VRADIO->JRADIO: R09 Telegram (repeated)
JRADIO->JC: R09 Message
JC->JRADIO: ack

IBIS->OBU: R09 Request\n(at same time as telegram)
OBU->RSU: CAM (> 1Hz)
RSU->JC: R09 Message
JC->RSU: processing
JC->RSU: processing
JC->JC: collecting requests
JC->RSU: granted
```

D.3 Sending R09 over SRM

```
participant IBIS as IBIS
participant vehicle radio\n(analog or digital) as VRADIO
participant OBU as OBU
participant RSU as RSU
participant junction radio\n(analog or digital) as JRADIO
participant junction controller as JC

IBIS->IBIS: reach reporting point
IBIS->VRADIO: R09 Telegram
VRADIO->JRADIO: R09 Telegram (repeated)
JRADIO->JC: R09 Message
JC->JRADIO: ack

IBIS->OBU: R09 Request\n(at same time as telegram)
OBU->RSU: SRM (repeated)
RSU->OBU: SSM ("received")
RSU->JC: R09 priority request
```

```
RSU->OBU: SSM ("requested")
OBU->IBIS: R09 Response ("requested")
JC->RSU: processing
RSU->OBU: SSM ("processing")
OBU->IBIS: R09 Response ("processing")
JC->JC: collecting requests
JC->RSU: processing
RSU->OBU: SSM ("processing")
OBU->IBIS: R09 Response ("processing")
JC->JC: collecting requests
JC->RSU: granted
RSU->OBU: SSM ("granted")
OBU->IBIS: R09 Response ("granted")
```

D.4 Sending SRM with ETA

```
participant IBIS as IBIS
participant OBU as OBU
participant RSU as RSU
participant junction controller as JC

IBIS->IBIS: operational Logon\n(path known)
IBIS->OBU: Path
JC->RSU: signal phase update
IBIS->IBIS: track location\n(along path)
IBIS->OBU: Path Location
OBU->OBU: process location\n(no junction upcoming -> do nothing)
RSU->OBU: MAP (> 1 Hz)
OBU->IBIS: Intersection Map
IBIS->IBIS: track location\n(along path)
IBIS->OBU: Path Location
OBU->OBU: process Location\n(junction upcoming -> calculate ETA)
OBU->IBIS: Intersection Status
OBU->RSU: SRM (including ETA)
RSU->JC: priority request (including ETA)
RSU->OBU: SSM ("requested")
JC->JC: collecting requests
JC->RSU: signal phase update
RSU->OBU: SPAT (> 1 Hz)
OBU->IBIS: Intersection Phase
JC->RSU: priority granted
RSU->OBU: SSM ("granted")
OBU->IBIS: Intersection Status

JC->RSU: signal phase update
RSU->OBU: SPAT (> 1 Hz)
OBU->IBIS: Intersection Phase

IBIS->IBIS: track location\n(along path)
IBIS->OBU: Path Location
OBU->OBU: process Location\n(junction upcoming -> recalculate ETA)
OBU->RSU: SRM (updated ETA)
RSU->JC: priority request (including ETA)
RSU->OBU: SSM ("requested")
JC->JC: collecting requests
JC->RSU: signal phase update
RSU->OBU: SPAT (> 1 Hz)
OBU->IBIS: Intersection Phase
```