

1 Data and messages description (Messages Catalogue)

1.1 Introduction

This Annex contains the data format and messages catalogue of the data transmitted on the Vehicle Ad-Hoc Network (VANET). The protocol is referred in the following as European ITS VANET Protocol (EIVP)

The protocol is intended to support the transmission of data among the stations (referred also as nodes) of an Ad hoc network using IEEE.802.11p as Physical and MAC ISO layers and covers the Network and Higher ISO layers.

The description is made using the ASN.1 syntax.

The general structure of an EIVP message is reported in graphical form in the following picture

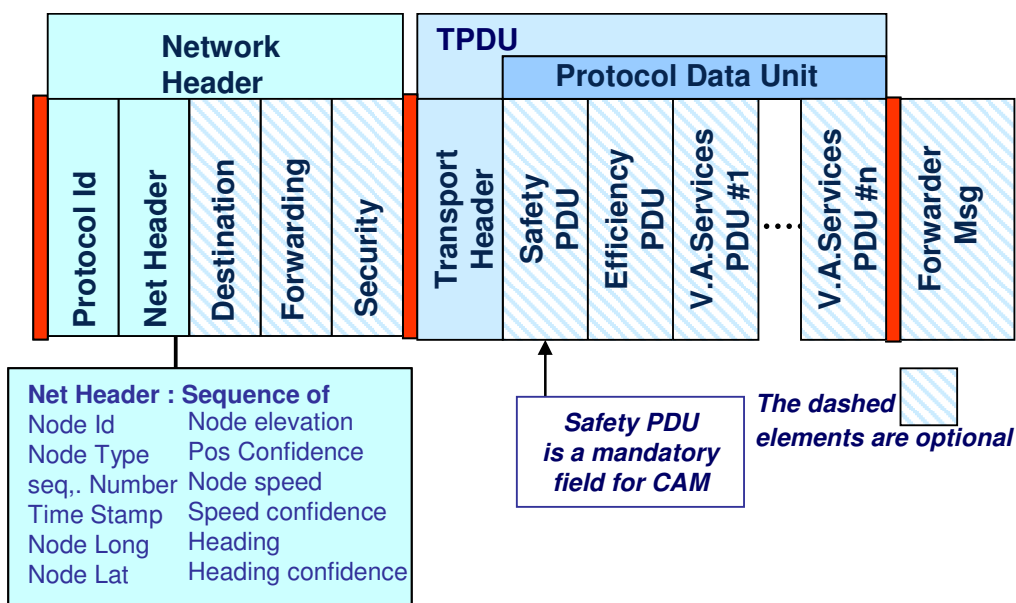


Fig. A11-1 EIVP message structure.

As shown in the picture two main blocks are considered:

- The Network Header
- The Transport Protocol Data Unit

The content and the structure of the messages are derived from the activities in C2C-CC and SAFESPOT and are extended to manage a more general class of services for which a general structure is provided and the content is not yet defined.

Currently no specific Efficiency PDU is specified. The Efficiency applications are supposed to use the information contained in the Safety PDU or messages using the same structure of the Value Added Services. This more general structure is the ServicePDU.

1.2 The General structure of the VANET messages

Here in the following the general structure of a message is reported using the ASN.1 syntax. As a general remark the ASN.1 syntax is used, but, to improve readability, sometimes formal rules are not respected. This means that some minor rework should be done in order to have a formally correct document to be used as input for an ASN.1 compiler.

```

EIVPMsg ::= SEQUENCE {
    protocolID          Protocol Identifier,
    originator          NetOHeader,
    destination        Destination OPTIONAL,
    forwarder          NodeId OPTIONAL,
                       -- Identifier of the last forwarding node
    security            NetSEC OPTIONAL,
    transport          SEQUENCE {
        source port    SrcPort,
        destinationPort DestPort
    } OPTIONAL,
    msgType            MsgType,
    genPDU             GenPDU OPTIONAL,
    servicesPDUS      SET (SIZE(0..31)) OF ServicePDU OPTIONAL,
                       -- A message without PDU is possible.
    forwarderMsg      ForwarderMsg OPTIONAL
}

```

```

GenPDU ::= CHOICE {
    coopAwareness      [CAMid] CamPDU,
    /* The Cooperative Awareness Message is used as substitute of the Network Layer Beacon.
    Currently the Default Frequency is 2Hz. */
    decentralizedSituation [DENid] DecentralizedSituationPDU,
    genSafety           [PERid] GenSafetyPDU,
    serviceInit        [ServInitId] ServicePDU,
}

```

```

ForwarderMsg ::= SEQUENCE { /* the forwarder message may be any complete message not
containing any forwarded message */
    protocolID          Protocol Identifier,
    originator          NetOHeader,
    destination        Destination OPTIONAL,
    security            NetSEC OPTIONAL,
    transport          SEQUENCE {
        source port    SrcPort,
        destinationPort DestPort
    } OPTIONAL,
    msgType            MsgType,
    GenPDU             GenPDU, OPTIONAL,
    servicesPDUS      SET OF(SIZE(0..31)) ServicePDU, OPTIONAL
                       -- A message without PDU is possible.
}

```

1.3 PDU of VANET messages set

/ This section contains the general structure of all the types of protocol data unit used by the different applications. CAMPDU and DecentralizedSituationPDU are derived from the version currently under discussion in the C2C-CC. */*

```

CampPDU::= CHOICE { /* CAMPDU is a message send by any node participating to the road safety
application and is used as a substitute of the Network Layer Beacon. Two different contents are
possible depending on the type of station ( Vehicle or Road Side Unit(RSU)) */

```

```

vehicleCAMPayload      VehicleCAMPayload,
rsuCAMPayload          RSUCAMPayload
}

```

```

DecentralizedSituationPDU ::= SEQUENCE {
    actionID             INTEGER,
-- Unique identifier of information about a situation from one source
    dataVersion          INTEGER (0..255),
-- Version of message indicating updates from same origin; 255: revocation
    generationTime       TimeStamp,
-- time of vehicle encountering the start location of the hazard
    expiryTime           TimeStamp,
-- time when message should be deleted from all databases and sending queues
    situationLon         Longitude,
-- longitude of reference position for situation. Also used for definition of distribution area
    situationLat         Latitude,
-- latitude of reference position for situation. Also used for definition of distribution area
    situationAlt         Elevation,
-- altitude of reference position for situation. Also used for definition of distribution area
    destinationArea     CHOICE {
        ellipse          [0] EllipLocData,
-- describes the destination area with points or distances values when the required destination area is
-- in the shape of ellipse
        circle           [1] CircLocData,
-- describes the destination area with points or distances values when the required destination area is
-- in the shape of circle
        rectangle        [2] RectLocData,
-- describes the destination area with points or distances values when the required destination area is
-- in the shape of rectangle
        ...,
    },
    reliability          INTEGER (0..100),
--probability of hazard information to be true
    isNegation          BOOLEAN,
-- negates the existence of a situation type at the given location
    situation            Situation,
-- container with situation description, incl. type
    severity             Severity,
-- describe the safety impact level of the detected situation e.g. very dangerous, dangerous,
-- informative
    location             LocationRef,
-- container with location description of the situation
    security             INTEGER -- TBD
}

```

```

GenSafetyPDU ::= SEQUENCE OF SEQUENCE{
/* A generic Safety PDU consist of a data payload formed by a tagged list of items chosen from
the list of possible items */
    tag                 StdVANETDataTagList,
    length              INTEGER(0..255),
    value               StdVANETDataValueList
} -- variable lenght

```

ServicePDU ::= SEQUENCE {

/ The Service PDU is used to announce, request, acknowledge and response to services
Service Identifier may range from 1 byte to 4 bytes. The extension is announced by the most
significant bit of the first byte. */*

```

    serviceId          INTEGER(0..127),
    idExtensionFlag    ENUMERATED {
                        (0) -- No Id Extension
                        (1) -- Id Extension
                        },
    idExtension         INTEGER (0..(2^24-1)) OPTIONAL,
    servicePayload     ServicePayload    OPTIONAL,
    -- The content is service dependent --
    localAddress       LocalAddress      OPTIONAL, -- TBD
    ServiceChannel     ServiceChannel    OPTIONAL, -- TBD
    channelInfo        ChannelInfo       OPTIONAL -- TBD
}

```

1.4 DATA FRAME set

/* In this section the set of DATA FRAME (sequence of elementary data – DATA ELEMENTS) are here reported. */

```

NetOHeader ::= SEQUENCE {
    nodeID           NodeID,           -- 8 bytes identifier
    nodeType         NodeType,
    sequenceNumber  INTEGER (0..255),
    timeStamp        VanetTimeStamp,
    nodePosition    SEQUENCE {
        nodeLongitude Longitude,
        nodeLatitude  Latitude,
        nodeElevation Elevation,
        positionConfidence Confidence OPTIONAL
    },
    speedModule      SEQUENCE {
        speed          SpeedModule,
        speedConfidence Confidence OPTIONAL
    },
    heading          SEQUENCE {
        heading        Heading,
        headingConfidence Confidence OPTIONAL
    },
    priority         Priority
    txPower          TxPower
}

```

```

VehicleCAMPayload ::= SEQUENCE {
    vehicleType      VehicleType,
    vehicleSize      VehicleSize,
    longitudinalAcceleration LongitudinalAcceleration,
    yawRate          YawRate,
    accelerationControl AccelerationControl,
}

```

```

    exteriorLights      ExteriorLights,
    taggedList          CAMTaggedList    OPTIONAL,
    satelliteData       SatelliteRawDataInBeacon OPTIONAL
    /*Ego Satellite Raw Data exchanged to be used for improving the relative positioning
    measurement among stations */
}

```

```

RSUCAMPayload::= SEQUENCE {
    nearestRSU          NearestRSU,
    linkType            LinkType,
    temperature         Temperature,
    visibilityRange     VisibilityRange,
    trafficDensity      TrafficDensity,
    weather             Weather,
    taggedList          CAMTaggedList    OPTIONAL,
    satelliteData       SatelliteRawDataInBeacon OPTIONAL
}

```

```

Destination::= SEQUENCE {
    /*In case the message is broadcasted the address is
        {routingType          →      RtBroadcast
        destinationDescription →      AdrNone
        timeToLive           →      xxx
    }*/
    routingType          ENUMERATED {
        Unknown          (0),
        RtBroadcast      (1),
        RtUnicast        (2),
        RtGeoUnicast     (3),
        RtGeoAnycast     (4),
        RtGeoBroadcast   (5),
        RtStoGeoUnicast  (6),
        RtStoGeoAnycast  (7),
        RtStoGeoBroadcast (8)
    }
    destinationDescription ENUMERATED {
        Unknown          (0),

```

```

        AdrNone      (1),
        AdrNode      (2),
        AdrGeo       (3),
        AdrNodeGeo   (4)
    }
    timeToLive       INTEGER (0..255),           /*LSB is 1s*/
    period           PeriodicMessagePeriod,   /*65535 value is
                                                    reserved for not
                                                    periodic
                                                    messages*/

    adrGeo           Geoaddress      OPTIONAL, /* TBD */
    adrNode          NodeID             OPTIONAL
}

```

```

TxPower::= SEQUENCE {
    power           EmittedPower -- 5 bit
    antenna         AntennaType -- 3 bit
} -- 8 bits

```

```

CircLocData ::= SEQUENCE { /* destination area represented as a circle */
    centerLon      Longitude,
    centerLat      Latitude,
    radius         Range
}

```

```

CAMTaggedList::= SEQUENCE (SIZE(0..32)) OF SEQUENCE {
/* each Data element has a tag number and may be inserted in the payload. The
StdVANETDataTagList include all the elements and frames defined in this document */
    tag           StdVANETDynamicDataTagList,
    length        INTEGER(0..255),
    value         StdVANETDataValueList
}

```

```

MessageTaggedList::= SEQUENCE {
/* each Data element has a tag number and may be inserted in the payload. The
StdVANETDataTagList include all the elements and frames defined in this document */
    tag           StdVANETDataTagList,
    length        INTEGER(0..255),

```

```

value          StdVANETDataValueList
}

```

```

StdVANETDataValueList ::= CHOICE {
    vehiclevalue          StdVANETVehicleDataValueList,
    rsuvalue              StdVANETRSUDataValueList
} -- variable length

```

```

NearestRSU ::= SEQUENCE (SIZE (4)) OF Position2D

```

```

LinkType ::= ENUMERATED {
    unknown                (0),
    urbanIntersection-D    (1),
    nonUrbanIntersection-D (2),
    motorway(2x2)-D        (3),
    motorway(2x3)-D        (4),
    motorway(3x3)-D        (5),
    motorway(3x4)-D        (6),
    motorway(4x4)-D        (7),
    interUrbanTrunkRoad(2x2)-CB-D (8),
    interUrbanTrunkRoad(2x3)-CB-D (9),
    interUrbanTrunkRoad(3x3)-CB-D (10),
    slipRoad(1 lane)-D     (11),
    slipRoad(2 lane)-D     (12),
    ringRoad(2x2)-CB-D     (13),
    ringRoad(3x3)-CB-D     (14),
    ruralRoad(1x1)-CB-D    (15),
    ruralRoad(1x2)-CB-D    (16),
    ruralRoad(2x2)-CB-D    (17),
    urbanIntersection-ND   (31),
    non-urbanIntersection-ND (32),
    motorway-(2x2)-ND      (33),
    motorway (2x3)-ND      (34),
    motorway (3x3)-ND      (35),
    motorway (3x4)-ND      (36),
    motorway (4x4)-ND      (37),

```

```

interUrbanTrunkRoad(2x2)-CB-ND (38),
interUrbanTrunkRoad(2x3)-CB-ND (39),
interUrbanTrunkRoad(3x3)-CB-ND (40),
slipRoad(1 lane)-ND (41),
slipRoad(2 lane)-ND (42),
ringRoad(2x2)-CB-ND (43),
ringRoad(3x3)-CB-ND (44),
ruralRoad(1x1)-CB-ND (45),
ruralRoad(1x2)-CB-ND (46),
ruralRoad(2x2)-CB-ND (47)
}

```

/ D = draining asphalt, ND = non draining, CB = central barrier, NCB = no central barrier*/*

```

AccelerationControl ::= BIT STRING {
    brakePedal (0), B'0000-0001
    gasPedalAct (1), B'0000-0010
    cruiseControl (2), B'0000-0100
    adaptiveCruiseControl (3), B'0000-1000
    limiter (4), B'0001-0000
    ....
}

```

```

VehicleType ::= SEQUENCE {
    vehicleCluster VehicleDescription,
    vehicleAttribute VehicleAttribute
}

```

```

VehicleMass ::= SEQUENCE {
    mass INTEGER (0..255), -- LSB defined by Unit, Value '0' → Data not available
    unit ENUMERATED {
        (0), -- LSB unit 10Kg
        (1), -- LSB unit 25Kg
        (2), -- LSB unit 100Kg
        (3) -- LSB unit 250Kg
    }
}

```

```

}
/* Example → With an LSB of 25 kg, this produces a max range of 6375kg. Mass should reflect
current gross mass of vehicle and contents if known, otherwise an average laden value should be
established. */
-----
LocationRef::= CHOICE{
    trace [0] SEQUENCE SIZE (0..16) OF ShortPosition2D,
    -- list of waypoints leading to the situation position the first
    ...
}
-----
Position2D ::= SEQUENCE {
    long2D          LongPosition,
    lat2D           LatPosition
}
-----
ShortPosition2D ::= SEQUENCE {
    shortLon       ShortLongitude,
    shortLat       ShortLatitude
}
-----
Speed ::= SEQUENCE { /* speed is represented as a vector by (module and direction(heading)).
Used for speeds elaborated by the Positioning system. */
    speedModule    SpeedModule,
    heading        Heading
}
-----
VehicleSize ::= SEQUENCE {
    width          VehicleWidth,
    length         VehicleLength,
}
-----
BrakeSystemStatus ::= SEQUENCE {
/* A single byte long data frame combining multiple related bit fields into one byte.*/
    pedalStatus    BOOLEAN, -- on=TRUE (pressed) off=FALSE (not pressed)
    brakeBoostApplied BrakeBoostApplied,
    cylinderPressure MasterCylinderPressure
}
-----

```

```

ExteriorLights ::= BIT STRING {
    allLightsOff          ( ),      B'0000-0000

    lowBeamHeadlightsOn  (0),      B'0000-0001
    highBeamHeadlightsOn (1),      B'0000-0010
    leftTurnSignalOn     (2),      B'0000-0100
    rightTurnSignalOn    (3),      B'0000-1000

    hazardSignalOn       ( ),      B'0000-1100

    automaticLightControlOn (4),    B'0001-0000
    daytimeRunningLightsOn  (5),    B'0010-0000
    fogLightOn            (6),      B'0100-0000
    parkingLightsOn       (7)      B'1000-0000
}

```

```

Environmental ::= SEQUENCE{
    temperat      Temperature,
    visibility    VisibilityRange,
    weather       Weather,
    road          RoadStatus,
    traffic       TrafficDensity,
    friction      FrictionRoadSurface, -- TPEG definition
    sideWind      Heading,
    surface       SurfaceCondition -- TBD
    twilight      Flag,
    nighttime     Flag,
    storm         Flag,
    rain          RainSensor,
    fog           Flag,
    snow          Flag,
    glare         Flag,
    sirenUse      SirenInUse,
    sunSensor     SunSensor
}

```

TrafficLightGeneralStatus ::=SEQUENCE (SIZE(1..32)) OF SEQUENCE {

/ It represents the status of traffic lights related to an RSU. The Sequence include the Direction, but very probably the TrafficLightId is enough. The direction may be useful for a traffic light not included in the Map. To be consolidated */*

```

    direction                Heading,
    trafficLightDirectionalStatus TrafficLightDirectionalStatus
}

```

TrafficLightDirectionalStatus ::=SEQUENCE {

```

    NumberOfLanes    INTEGER (0..7)
    SEQUENCE (SIZE(1..7)) OF SEQUENCE {
        LaneNumber        INTEGER (1..7)
        CurrentLightStauts TrafficLightStatusShort
        NextLightStatus   TrafficLightStatusShort
        TimeToNextstauts  VANETTimeStamp
    }
}

```

TrafficLightStatusShort ::= BITSTRING {

```

-- allLightsOff      ( ),   B'0000 0000  B'0000 0000
Circle               (0),   B'0000 0000  B'0000 0001
Left Arrow           (1),   B'0000 0000  B'0000 0010
Right Arrow          (2),   B'0000 0000  B'0000 0100
Strait Arrow         (3),   B'0000 0000  B'0000 1000
Pedestrian           (4),   B'0000 0000  B'0001 0000
Bicycle              (5),   B'0000 0000  B'0010 0000
LeftPT               (6),   B'0000 0000  B'0100 0000
RighPT               (7),   B'0000 0000  B'1000 0000
StraightPT           (8),   B'0000 0001  B'0000 0000
StopPT               (9),   B'0000 0010  B'0000 0000
TrianglePT           (10),  B'0000 0100  B'0000 0000
Red                  (11),  B'0000 1000  B'0000 0000
Yellow               (12),  B'0001 0000  B'0000 0000
Green                (13),  B'0010 0000  B'0000 0000
Blinking             (14),  B'0100 0000  B'0000 0000
}

```

TrafficLightStatus ::= SEQUENCE (SIZE(0...numberSignalGroups)) OF SEQUENCE {

id	SGID ,	<i>/* ID of Signal Group */</i>
colour	SGColour ,	
nextColour	SGColour ,	
colourResidualMin	SGResidTime ,	
colourResidualMax	SGResidTime ,	
status	SGstatus ,	
nextStatus	SGstatus ,	
statusResidualMin	SGResidTime ,	
statusResidualMax	SGResidTime	

}

GreenWave ::= SEQUENCE (SIZE(0...numberGreenWaves)) OF SEQUENCE{

exitLink	GWExitLink ,	<i>/* intersection related road ID, where Green Wave is valid for a leaving car, "0"(unknown) (3) */</i>
greenwaveSpeed	GWSpeed	<i>/*coordination speed in km/h, "255" (unknown) */</i>

}

PedestrianDetectorData ::= SEQUENCE (SIZE(0...numberDetPedestrian)) OF SEQUENCE {

id	DetID ,	<i>/* Intersection related ID of Pedestrian Detector. This ID has to be stored in the LDM and the according detection area. */</i>
status	DetStatus ,	<i>/* operatiing state of vehicle detector "0"(unknown) "1"(operating), "2"(out_of_order) */</i>
presence	DetPedPresence	

}

AggDetectorData ::= SEQUENCE (SIZE(0...NumberRoadDet)) OF SEQUENCE {

id	DetID ,	<i>/* Intersection related ID of road detector. This ID has to be stored in the LDM and the according detection area. */</i>
status	DetStatus ,	<i>/* operating state of vehicle detector "0"(unknown) "1"(operating), "2"(out_of_order) */</i>
aggregationIntervall	DetAgglInt ,	
flow	DetFlow ,	

```

occupancy      DetOcc,
truckShare     DetTruck,
speed          DetSpeed,
deviationSpeed DetDevSpeed
}

```

RawDetectorData ::= SEQUENCE (SIZE(0..NumberRoadDet)) OF SEQUENCE {

```

id          DetID,          /* Intersection related ID of road detector.
                        This ID has to be stored in the LDM and the according
                        detection area. */

status      DetStatus,    /* operating state of vehicle detector "0"(unknown)
                        "1"(operating), "2"(disordered) */

interval    DetInterval, /* in steps of 0.1 sec, 10 is 1 second.

state       DetState,     /* on/off at end of interval

occupancy   TimeOn,       /* in steps 0.01 sec., 100 is 1 second

speed       DetSpeed,     /* in steps 0.1 m/s., 10 is 1 second

class       VehClass,     /* (enumerator) */

count       DetCount     /* all pulses in interval, reference = end of pulse
}

```

ProloRequest ::= SEQUENCE {

```

severity      ProSeverity,
minDuration   ProMinDuration,
violatedSignalGroup ProViolatedSignalGroup,
exitLinkid    ProExitLinkID
}

```

Trajectory ::= SEQUENCE {

/ A trajectory is defined by an initial point (defined as the trajectory origin) and a sequence of way points defined with relative coordinates. Associated to the each point there is either the speed (the desired or maximum speed in that point) or the time (forecasted relative time respect to the origin in which the point should be reached).*

The trajectories are transmitted to the vehicles on the VANET by a Road Side unit in curve departure application. Different types of trajectories are defined internally in LDM referred to ego or other vehicles/*

```

numberOfPoints CntPoints, /* number of Way point of the Trajectory up to 128 */
timeFlag       Flag,      /* Y= trajectoryTime, N = trajectorySpeed */

```

```

trajectory origin      Position2D, /* initial position of the trajectory */
trajectorySpeed        TrajectorySpeed    OPTIONAL,
trajectoryTime         TrajectoryTime     OPTIONAL
}

```

ObstacleParameters ::= SEQUENCE {

/ Obstacle defined according to SAEJ2735. The Obstacle can be an object detected by on board or infrastructure sensors or a VANET node different from ego node. */*

```

iD                    ObstacleID,           -- In Case of a VANET Node the
                                                Nodeld Otherwise the ego node
                                                will use a local (LDM assigned)
                                                identifier

obstacleLongitude    LongPosition,         -- Position associated to
                                                geometrical center

obstacleLatitude     LatPosition,
obstacleAltitude     Elevation,
objectAge             INTEGER (0..65535)    -- units of 10/125 seconds
source                ObstacleSource,
confidence            ObstacleConfidence,
type                  ObstacleType,       -- to be better specified
width                 ObstacleWidth,
length                ObstacleLength,
height                ObstacleHeight,
weight                ObstacleWeight,
geographicArea        ObstacleGeographicArea, -- Road segment
lanesOccupied         ObstacleLanesOccupied,
timeStampComplete    VANETTimeStamp
}

```

TrajectorySpeed ::= SEQUENCE{ */*set of point and speeds */*

```

x-coordinate          CoordinateTrajectory, /* relative coordinates referred to transmitter
                                                position */

y-coordinate          CoordinateTrajectory,
speed                 SpeedModule         /* desired or max safe speed corresponding
                                                to the coordinates */

```

/ The range values are referred to the max value defined for the longitudinal speed of the vehicle*/*

```

}
```

```

TrajectoryTime ::= SEQUENCE{
    time                TimeStampRelative,
    x-coordinate        CoordinateTrajectory,
    y-coordinate        CoordinateTrajectory
}

```

```

UDPTimeStamp::=SEQUENCE {
    seconds             INTEGER (32bit),    -- the Linux-UTC representation reduced to 6
                                                bytes (ms. resolution)
    milliseconds       INTEGER (16bit)    -- (2^32 bits -1) seconds since 1/1/1970
                                                -- milliseconds
}

```

/ In case of not available or not valid time stamp, both fields must be set to 0 */*

```

SatelliteUTCTimeStamp::= SEQUENCE {
    hour               INTEGER(0..23),
    minute             INTEGER(0..59),
    seconds            REAL                -- Float single precision. Unit is 1s.
}

```

```

TimeStamp ::= INTEGER (0..4294967295) -- UTC seconds since 1.1.1970 00:00:00.000

```

```

VANETTimeStamp ::= SEQUENCE{
    seconds            INTEGER (32bit),    /* the Linux-UTC representation reduced to 6
                                                bytes (ms.resolution) */
    milliseconds       INTEGER (16bit)    -- (0..4294967295) seconds since 1/1/1970
                                                -- milliseconds
}

```

-- In case of not available or not valid time stamp, both fields must be set to 0

```

SatellitesRawData ::= SEQUENCE {
    posSource          PosSource,
    /* positioning sensors used for the estimation of the position of the transmitting node */
    satelliteCnt       INTEGER (0..8),
    /*0 = no data otherwise number of satellites seen by the transmitting node */
}

```

```
satelliteData      SEQUENCE (SIZE(0..8)) OF SatelliteRawDataBeacon
                   OPTIONAL
```

```
/* It contains the raw data for each GPS satellite that the receiver can view*/
```

```
}
```

```
-----
```

```
SatelliteRawDataInBeacon ::= SEQUENCE {
    satelliteFixed      SatelliteFixed,
    numOfSatellitesUsed  INTEGER (4..255),
    variableSatelliteData SEQUENCE (SIZE(4..12)) OF {
        eachSatelliteData EachSatelliteData
    }
}
```

```
-----
```

```
SatelliteFixed ::= SEQUENCE{
    timestamp      SatelliteUTCTimeStamp,
    x              REAL,      -- Float single precision. Unit is 1cm.
    y              REAL,      -- Float single precision. Unit is 1cm.
    z              REAL,      -- Float single precision. Unit is 1cm.
    latitude       REAL,      -- Float single precision. Unit is 1degree.
    longitude      REAL,      -- Float single precision. Unit is 1degree
    altitude       REAL,      -- Float single precision. Unit is 1cm.
    vx            REAL,
    -- Float single precision. Unit is 1cm/s. ECEF x velocity
    vy            REAL,
    -- Float single precision. Unit is 1cm/s. ECEF y velocity
    vz            REAL,
    -- Float single precision. Unit is 1cm/s. ECEF z velocity
    ve            REAL,
    -- Float single precision. Unit is 1cm/s. NED North velocity
    vn            REAL,
    -- Float single precision. Unit is 1cm/s. NED East velocity
    vd            REAL,
    -- Float single precision. Unit is 1cm/s. NED Down velocity
    hdop          INTEGER(0..65535), -- Horizontal Dilution of Precision
    gdop          INTEGER(0..65535), -- Geometric Dilution of Precision
    vdop          INTEGER(0..65535), -- Vertical Dilution of Precision
    tdop          INTEGER(0..65535), -- Time Dilution of Precision
```

```

    groundspeed      REAL,           -- Float single precision. Unit is 1cm/s.
    heading          REAL           -- Float single precision. Unit is 1degree
}

```

```

EachSatelliteData::= SEQUENCE {
    usedSatelliteID    INTEGER(0..255),
    elevation          INTEGER(0..255),  -- LSB is 1 degree
    azimuth            INTEGER(0..65535), -- LSB is 1 degree
    residualPseudorange REAL,           -- Float single precision. Unit is 1cm
    pseudorange        REAL,           -- Float double precision. Unit is 1m
    carrierToNoiseRatio INTEGER(0..255) -- LSB is 1 dB
}

```

```

SatelliteRawDataBeacon ::= SEQUENCE {
    pRN                SatelliteID,      -- GPS Satellite ID
    sT                 INTEGER (0..255),    -- Signal Strenght
    pR                 PseudoRange
}

```

```

SystemsOnBoard ::= SEQUENCE {
    programmedSpeedControlValue ProgrammedSpeedControlValue,
    abs                         AbsStatus,
    esp                         StabilityControlStatus,
    airbagOrCrash              AirbagOrCrash,
    tankFilingLevel            TankFilingLevel,
    doorStatus                 DoorStatus, -- all doors closed = TRUE
    tirePressure                TirePressure
}

```

```

GeographicArea::= SEQUENCE{
    /* a geographical area is defined as a polygon described by a set of points corresponding to its
    vertices. Points position is expressed by latitude and longitude */
    pointCnt    INTEGER(0..32),  -- Number of points describing the area
    pointsList  SEQUENCE (SIZE(pointCnt)) OF Position2D
}

```

}

RsuHMIMessage::= SEQUENCE {

/ this is the description of a specific frame used by infrastructure to communicates Coded Icons and/or text to vehicles. The Infrastructure node sends this info embedded in a periodic or event message. The data structure used is the tag – value sequence */*

```

msgId          INTEGER (0..65535),
cntVehicleType INTEGER (0..255),
vehicleType    SEQUENCE (SIZE(0..63)) OF VehicleDescription,
application    INTEGER (0..255),
-- code of the application which generated the message
usecase        INTEGER (0..255),
-- specific use case in which the application was generated
status         ENUMERATED {
/* classification according to the SAFESPOT safety margin applications rules*/
                comfort      (0),
                safety        (1),
                critical      (2)
            },
secondaryActor ObstacleType,
-- type of secondary actor involved ( SAFESPOT definition)
hmiChannel     ENUMERATED {
                visual (0),
                audio (1),
                haptic (2)
                ...
            },
secondaryModality BOOLEAN, -- HMI modality primary = FALSE, secondary = TRUE */
priority       INTEGER(0..255), -- HMI priority
timeToEvent    INTEGER(TBD1..TBD2), -- range and resolution to be defined
distanceToEvent INTEGER(TBD3..TBD4), -- range and resolution to be defined
iconHearcon    INTEGER(0..1024),
-- Coded images or sounds (0 = no icon or hearcon)
textMessage    UTF8string(SIZE (0..63)),
-- up to 63 characters strings allowed . If the first character is 0(NULL) no string is attached.
timeValidity   SEQUENCE {

```

```

        initialtime    VANETTimeStamp,
        finaltime      VANETTimeStamp
    }
    geoValidity        GeographicArea -- polygon defined by the position of its vertex
}
-----
-- NetSEC TBD ( network security not yet defined)
-----

ProtocolId ::= SEQUENCE {
    protocolType      INTEGER (0..15),
    protocolVersion   INTEGER (0..14, ..., 15..270)
}
-----

StdVANETVehicleDataValueList::= CHOICE {
/* list of Data than can compose the variable part of the payload associated to a VANET message sent
by a vehicle. Any combination of these data is possible. */
    brakeSystemStatus    BrakeSystemStatus,
    environmental         Environmental,
    obstacleParameters    ObstacleParameters,
    systemsOnBoard        SystemsOnBoard,
    temperature           Temperature,
    visibilityRange        VisibilityRange,
    weather               Weather,
    trafficDensity         TrafficDensity,
    frictionRoadSurface    FrictionRoadSurface,
    rainSensor            RainSensor,
    sirenInUse            SirenInUse,
    sunSensor             SunSensor,
    vehicleMass           VehicleMass,
    meanSteerAngle        MeanSteerAngle,
    steeringWheelAngle    SteeringWheelAngle,
    steeringWheelAngleRateOfChange SteeringWheelAngleRateOfChange,
    masterCylinderPressure MasterCylinderPressure,
    gasPedal              GasPedal,
    tiltSensor            TiltSensor,
    lightbar-In-Use       LightbarInUseSAE,
    gearStatus            GearStatus,
    brakeBoostApplied     BrakeBoostApplied,

```

```

windscreenWiper      WindscreenWiper,
wiperRate            WiperRate,
programmedSpeedControlValue ProgrammedSpeedControlValue,
airbagOrCrash        AirbagOrCrash,
stabilityControlStatus StabilityControlStatus,
tirePressure         TirePressure,
tankFilingLevel      TankFilingLevel,
typeOfGoods          TypeOfGoods
}

```

StdVANETRSUDataValueList::= CHOICE {
/ list of Data than can compose the variable part of the payload associated to a VANET message sent by a vehicle. Any combination of these data is possible. */*

```

Environmental        Environmental,
TrafficLightStatus   TrafficLightGeneralStatus,
Trajectory           Trajectory,
ObstacleParameters   ObstacleParameters,
Temperature          Temperature,
VisibilityRange       VisibilityRange,
Weather              Weather,
TrafficDensity        TrafficDensity,
FrictionRoadSurface   FrictionRoadSurface,
RainSensor           RainSensor,
SirenInUse           SirenInUse,
SunSensor            SunSensor,
RoadID               RoadID,
RoadType             RoadType,
TLCStatus            TLCStatus
}

```

1.5 /* data Elements */

```

AbsStatus ::= ENUMERATED {
notEquipped (0),  -- Not equipped
off          (1),  -- Off
on           (2),  -- On
engaged     (3)   -- Engaged
}

```

AirbagOrCrash ::= BIT STRING {

/ A list of Airbag or Crash detected is used . In case of vehicles which have no details about the airbag the B'11111111' is used.*/*

```

    -- noCrashDetected      ( ),      -- B'0000-0000
    frontPtCrashDetected    (0),      -- B'0000-0001
    front1LevelCrashDetected (1),      -- B'0000-0010
    front2LevelCrashDetected (2),      -- B'0000-0100
    psngrSideCrashDetected  (3),      -- B'0000-1000
    driverSideCrashDetected (4),      -- B'0001-0000
    rearCrashDetected       (5),      -- B'0010-0000
    rolloverDetected        (6),      -- B'0100-0000
    pedestrianCrashDetected (7),      -- B'1000-0000
    -- anyAirbagOrCrashDetected ( )    -- B'1111-1111'
    -- any or all airbag exploded or crash anyhow detected
}

```

Angle ::= INTEGER (0..360000) *-- LSB units of 0.001 deg (19 bits)*

AntennaType ::= ENUMERATED {

*/*Indication of the antennaType used for VANET communication*/*

```

    unknown      (0),
    omnidirectional (1)

```

*/*the remaining fields aren't yet filled*/*

}

BeaconPeriod ::= INTEGER(0..255) *-- LSB units of 50ms*

BrakeBoostApplied ::= ENUMERATED {

```

    notEquipped (0),
    off          (1),
    on           (2)
}

```

BrakePedalStatus ::= BOOLEAN *-- pressed = TRUE, not pressed = FALSE*

CntPoints ::= INTEGER (0..128) *-- Number of points (8 bits)*

Confidence ::= INTEGER (0..15)

/ The data element provides the symmetric interval of 95% confidence level for a current reported value. The confidence limits of the interval are calculated based on the Granularity of the*

*corresponding measurement data element according to: Limit = \pm LSB_Value * 2 ^ Confidence. 15 is set if no other value is available.*

*The use of 95% confidence intervals and the logarithmic scale is compliant to SAE J2735. */*

```

Coordinate ::= INTEGER (-108..108)           -- LSB units of 0.01 m (27 bits)
CoordinateTrajectory ::= INTEGER (-10000..10000)
/* LSB units of 0.1 m (15 bits) +/-1 Km range; relative coordinates referred to transmitter position */
DetAggInt ::= INTEGER (0..65535)             /* LSB units of 1 s; aggregation interval in seconds */
DetCount ::= INTEGER (0..255)               /* all pulses in interval, reference = end of pulse*/
DetDevSpeed ::= INTEGER (0..65535)         /* Standard deviation of Speed of the vehicles km/h,
"65535"(unknown) */
DetFlow ::= INTEGER (0..65535)             /* Traffic flow in vehicles/hour, "65535"(unknown) */
DetID ::= INTEGER (0..255)                  /* Intersection related ID of the detector; vehicle as
well pedestrian*/
DetInterval ::= INTEGER (0..65535)         /* time interval in steps of 0.1 sec, 10 is 1 second*/
DetOcc ::= INTEGER (0..65535)              /*Occupancy time in % of interval time,
"65535"(unknown) */
DetPedPresence ::= INTEGER (0..255)        /* "0" (nobody there), "1" (presence detected), "255"
(unknown)*/
DetSpeed ::= INTEGER (0..255)              /* vehicle speed in steps of 0.1 m/s., 10 is 1 second*/
DetState ::= BOOLEAN                       /* at end of interval; FALSE = off, TRUE = on*/
DetStatus ::= ENUMERATED {
    unknown           (0),
    operating         (1),
    disordered        (2)
}
DetTruck ::= INTEGER (0..65535)           /* Percentage of trucks, "65535"(unknown) */
DoorStatus ::= BIT STRING {
    -- allDoorsClosed      ( ),      B'0-0000
    driverDoorOpen        (0),      -- B'0-0001
    passengerDoorOpen     (1),      -- B'0-0010
    leftRearDoorOpen      (2),      -- B'0-0100
    rightRearDoorOpen     (3),      -- B'0-1000
    rearHatchOpen         (4),      -- B'1-0000
    -- allDoorsOpen        ( )      B'1-1111
}

DriverTiredness ::= INTEGER (0..255)      -- To be defined
DstPort ::= INTEGER (0..65535)           -- Destination port for routing of messages to applications
Elevation ::= INTEGER (-5000..32767)      /* 0,2m. LSB with a 1Km neg offset (16 bits). It
represents the elevation respect to the sea level. Actual range (-1000..6534) m. The value +32767
means value not available. */

```

EmittedPower::= INTEGER (0..31) */* LSB unit are dBm (5 bits). The value represent the transmitted power (EIRP) in dBm, i.e. the device output power increased by max antenna gain and decreased by cable losses. TX power is in the range of 0..31 dBm. (entries 32..63 are reserved)*/*

Forwarder::= INTEGER (0..(2³²-1)) *-- Address of the last forwarder node*

LongitudinalAcceleration ::= INTEGER (-2000..2000) */* LSB units of 0.01 m/s² (12 bits). Actual range: +/- 2G range. */*

SpeedModule ::= INTEGER (-32768..32767)

/ Units of 0.01 m/s (16 bits). Actual range (-327,68..327,67) m/s. Speed = 327,65 means no speed available. Negative values imply the vehicle is moving in reverse. Data not available or not valid expressed through the use of Variance */*

YawRate ::= INTEGER (-32768..32767)

/ LSB units of 0.01 degrees/s -actual range (-327,68..327,67). Yaw rate = 327,67 means no yaw rate available. Vehicle yaw-rate is positive when the vehicle is turning counter clockwise.*/*

Flag ::= BOOLEAN *-- Y/N (1 bit)*

FrictionRoadSurface ::= ENUMERATED {

/ The road surface friction is described in 6 bits. The MSB indicates if a significant condition occurs. If '1' there are normal condition (32) and the following bits are not significant. If '0' a special condition occurs and the following 5 bits are significant (TPEG). All bit set to '1' means unknown conditions. */*

petrolSpillage	(0),
oilSpillage	(1),
dieselSpillage	(2),
mud	(3),
looseChippings	(4),
leaves	(5),
snox	(6),
deepSnow	(7),
packedSnow	(8),
freshSnow	(9),
meltingSnow	(10),
snowDrifting	(11),
sleet	(12),
ice	(13),
icePatches	(14),
blackIce	(15),
floodWater	(16),
burstWaterMain	(17),
sewerOverflow	(18),
wornOutSurface	(19),
flashFlood	(20),
polixhedSurface	(21),
surfaceWater	(22),

```

normal          (32),
unknown        (63)
}

```

GasPedal ::= INTEGER (0..200)

/ LSB units of 0.5%. The position of the gas pedal in the vehicle, expressed in units of 0.5 percent of range of travel, unsigned.*/*

GearStatus ::= INTEGER (-10..50)

/ LSB units of 1. Indicates the gear: <0: reverse; 0: neutral; >0: gear number; +63 = not available */*

Heading ::= INTEGER (0..65535)

/ LSB of 0.005493247 degrees. The current heading of the vehicle, expressed in signed units of 0.005493247 degrees from North (such that 65,535 such degrees represent 360 degrees). North shall be defined as the axis defined by the WSG-84 coordinate system and its reference ellipsoid. Increasing when turning clockwise. Data not available or not valid expressed through the use of Variance */*

LateralAcceleration ::= INTEGER (-2000..2000)

/ LSB units of 0.01 m/s² (12 bits). Actual range: +/- 2G range. Positive when vehicle is turning left*/*

LatPosition ::= INTEGER (-720000000..720000000)

/ LSB = 1/8 micro degree. Actual range: (-90..+90) degrees.*

*Position of the geometrical centre (crossing point of rectangle diagonals) of the object. The geographic latitude of a node, expressed as a 31 bit value. Data not available or not valid expressed through the use of Variance */*

LightbarInUse ::= ENUMERATED {

```

notEquipped (0),
notInUse    (1),
inUse       (2),
reserved    (3)  -- for future use
}

```

/ Used to reflect any type or style of visual alerting when a vehicle is progressing and transmitting DSRC messages to others nearby vehicles about its path*/*

LongitudinalAcceleration ::= INTEGER (-2000..2000) -- LSB units of 0.01 m/s². Range: +/- 2g

LongPosition ::= INTEGER (-1440000000.. 1440000000)

/ LSB = 1/8 micro degree. Actual range: (-180..+180) degrees.*

Position of the geometrical centre (crossing point of rectangle diagonals) of the object. The geographic longitude of a node as a 32 bit value and with reference to the horizontal datum specified by horizontalDatum. Data not available or not valid expressed through the use of Variance/*

MasterCylinderPressure ::= ENUMERATED {

/ Is not a good indication of the intensity of a braking action (the value is depending on the type of braking system in the vehicle, that can change between different brands. Also; for example a truck does not have this parameter at all*/*

```

notEquipped (0),    -- Not Equipped
minPressure (1),    -- Minimum Braking Pressure
bkLvl-2      (2),
bkLvl-3      (3),
bkLvl-4      (4),
bkLvl-5      (5),
bkLvl-6      (6),
bkLvl-7      (7),
bkLvl-8      (8),
bkLvl-9      (9),
bkLvl-10     (10),
bkLvl-11     (11),
bkLvl-12     (12),
bkLvl-13     (13),
bkLvl-14     (14),
maxPressure (15)    -- Maximum Braking Pressure
} -- (4 bits)

```

MeanSteerAngle ::= INTEGER (-9000..9000) *-- LSB units of 0.01deg (15 bits)*

MsgType ::= ENUMERATED {

```

initService      (0),
ackService       (1),
reqService       (2),
resService       (3),

```

...

}

/ Identifies the type of message related to the scope of the message. All messages which are not requiring any service or acknowledge are "servicelit" type */*

NodeID ::= INTEGER (0..2⁶⁴-1) *-- Indicates a unique identifier on VANET*

NodeType ::= ENUMERATED {

```

vehicle          (0),
RSU              (1),

```

...

}

NumberOfLanes ::= INTEGER (-10⁶..10⁶)

NumberOfSatellites ::= INTEGER (0..8) -- *Number of Satellites that GPS receiver is receiving*

ObstacleConfidence ::= INTEGER (0..100) -- *0 value means obstacle disappeared*

ObstacleGeographicArea ::= INTEGER (0..255) */*The static map data are provided in segments */*

ObstacleHeight ::= INTEGER (0..60) -- *LSB units of 0.1 m*

ObstacleID ::= INTEGER (0.. 4294967295) -- *LSB units of 1*

ObstacleLanesOccupied ::= INTEGER (0..15) */* With this parameter we mean the lane where the obstacle is located, 0 means unknown, 1 means right lane.*/*

ObstacleLength ::= INTEGER (0..300) -- *LSB units of 0.1 m*

ObstacleSource ::= ENUMERATED {
 vanet (0),
 camera (1),
 radar (2),
 laser (3),
 ...
 }

ObstacleType ::= INTEGER (0..255) -- *TBD*
*/*It can be a vehicle (car, truck, PTW), a VRU (pedestrian, bicycle etc.) or maybe a rock, a box that fell from a truck etc. may be an animal (cow, sheep,etc.), a fallen tree, debris, spillage: TPEG rtm12 (object) rtm19&rtm20 (people) rtm21&rtm22&rtm23 (animal) */*

ObstacleWeight ::= INTEGER (0..100000) -- *LSB units of 20 Kg*

ObstacleWidth ::= INTEGER (0..50) -- *LSB units of 0.1 m*

OtherSpecificFeature ::= INTEGER (0..255) -- *LSB units of 1*

ParkingBrakeStatus ::= ENUMERATED {
 off (1),

```

on      (2)
}

```

PeriodicMessagePeriod ::= INTEGER(0..65535) */* LSB units of 100ms. 65535 value is reserved for not periodic messages. */*

Priority ::= ENUMERATED {

unknown	(0),
beacon	(1), -- <i>normally associated to CAM</i>
emergency	(2),
high	(3),
medium	(4),
low	(5),
...	

}

--VANET Message Priority

Probability ::= INTEGER (0..100)

ProgrammedSpeedControlValue ::= INTEGER (0..6000)
/ LSB 0.01 m/s. This is used to specify the desired speed of a Cruise Control system */*

PseudoRange ::= INTEGER (0.. 4294967295) *-- LSB units of 0,01 m. Range: ~43 Km*

RainSensor ::= ENUMERATED {

none	(0),
lightMist	(1),
heavyMist	(2),
lightRainOrDrizzle	(3),
rain	(4),
moderateRain	(5),
heavyRain	(6),
heavyDownpour	(7)

}

/ A general sensor of rain intensity which requires further interpretation by the OEM for precise semantic meaning. The "Rain Sensor" Probe Data Element is intended to inform Probe Data Users as to how hard it was raining/snowing in the area the vehicle was traveling at the time the Probe Data snapshot was taken. The value of the Rain Sensor data element ranges from 0-7, with 0 indicating "No Rain/Snow", 1 indicating "Light Mist", and 7 indicating "Heavy Downpour". This information could be sent to vehicles approaching the area to warn drivers of raining/snowing conditions ahead or it could provide Traffic Operation Centers with locations most likely in need of a snowplow.*/*

Range ::= INTEGER(0..255) -- Range of a circular destination area LSB: 25 m

RoadType ::= INTEGER (0..15) -- TBD. Highway, main road, secondary road, urban, extra-urban, rural*/

SatelliteID ::= INTEGER (1..32) -- Id of satellite in GPS constellation

SensorConfidence ::= INTEGER (0..255)

/ The confidence level of the detected object. The value is in range 0...100. 0 means unknown and 100 full confidence. In addition there are two status type values: 254: not updated, 255: not valid */*

SGColour ::= ENUMERATED {

unknown	(0),
dark	(1),
red	(2),
redAmber	(3),
green	(4),
amber	(5),
greenBlinking	(6),
amberBlinking	(7)

}

SGID ::= INTEGER (0..255)

SGResidTime ::= INTEGER (0..255)

*/*Meaning of "min, max": Time coding*

min, max = "255" => no time value available

min = max < "255" => precise value of residual time (typically for steady fixed time control)

min < max < "255" => value range of residual time (typically for traffic actuated local control)/*

SGStatus ::= INTEGER (0....255)

/ "0"(unknown), "1"(NoRightofWay), "2"(RightofWay); if actual Signal_State is RoW => value predicts time to end of RoW.*

Note: The forecasts can change from second to second (e.g. through PT priority, emergency vehicles)/*

SGType ::= ENUMERATED {

vehicle	(1),
vehicleLeft	(2),

```

    vehicleRight      (3),
    bus               (4),
    tram              (5),
    cyclist           (6),
    pedestrians       (7),
    diagonalBlinking (8),
    diagonalGreen     (9)
}
/*Maybe the chambers of the lights itself should be considered as well; Austria e.g. green blinking*/

```

ShortLatitude ::= INTEGER (0..65535)

/ reports the difference between Latitude of the point to be located (e.g. a waypoint of a trajectory) and the reference Latitude (e.g. the position of the transmitting station)*/*

ShortLongitude ::= INTEGER (0..65535)

/ reports the difference between Longitude of the point to be located (e.g. a waypoint of a trajectory) and the reference Longitude (e.g. the position of the transmitting station)*/*

SirenInUse ::= ENUMERATED {

/ A data element which is set if any sort of audible alarm is being emitted from the vehicle. This includes various common sirens as well as backup up beepers and other slow speed manoeuvring alerts.*

*Used to reflect any type or style of audio alerting when a vehicle is progressing and transmitting DSRC messages to others about its path. Intended to be used as part of the DSRC safety message for public safety vehicles operating in the area.**

```

    notEquipped (0),
    notInUse    (1),
    inUse       (2),
    reserved    (3)    -- for future use
} -- 2 Bits

```

Slope ::= INTEGER (0..63) *-- LSB units of 1*

SrcPort ::= INTEGER (0..65535) *-- Source port of the message (used by applications)*

StabilityControlStatus ::= ENUMERATED { *-- ESP*

```

    notEquipped (0),    -- B'00 Not equipped
    off         (1),    -- B'01 Off
    on          (2)     -- B'10 On or active
}

```

StdVANETDataTagList ::= ENUMERATED {

```

    unknown      (0),

```

accelerationControl	(1),
airbagOrCrash	(2),
brakeBoostApplied	(3),
exteriorLights	(4),
frictionRoadSurface	(5),
gasPedal	(6),
gearStatus	(7),
lightbarInUse	(8),
linkType	(9),
longitudinalAcceleration	(10),
masterCylinderPressure	(11),
meanSteerAngle	(12),
positionCovariance	(13),
programmedSpeedControlValue	(14),
rainSensor	(15),
sirenInUse	(16),
stabilityControlStatus	(17),
steeringWheelAngle	(18),
steeringWheelAngleRateOfChange	(19),
sunSensor	(20),
tankFilingLevel	(21),
temperature	(22),
tiltSensor	(23),
tirePressure	(24),
trafficDensity	(25),
typeOfGoods	(26),
vehicleElevation	(27),
vehicleSize	(28),
vehicleType	(29),
visibilityRange	(30),
weather	(31),
windscreenWiper	(32),
wiperRate	(33),
yawRate	(34)
brakeSystemStatus	(64),
environmental	(65),
rsuHmiMessage	(66),
vehicleHmiMessage	(67),

obstacleParameters	(68),
systemsOnBoard	(69),
trafficLightStatus	(70),
trajectory	(71),
vehicleMass	(72),
reserved	(255)

/ note: → currently the value used in SAFESPOT are reported */*

StdVANETDynamicDataTagList ::= ENUMERATED {

/ Tags of the of the Data than can compose the variable part of the payload associated to a VANET message. Any combination of these data is possible. If requested the addressed data may be updated by the Router before sending the message. Only data elements can belong to this enumerate, no data frame. */*

unknown	(0),
accelerationControl	(1),
airbagOrCrash	(2),
brakeBoostApplied	(3),
exteriorLights	(4),
frictionRoadSurface	(5),
gasPedal	(6),
gearStatus	(7),
lightbarInUse	(8),
linkType	(9),
longitudinalAcceleration	(10),
masterCylinderPressure	(11),
meanSteerAngle	(12),
positionCovariance	(13),
programmedSpeedControlValue	(14),
rainSensor	(15),
sirenInUse	(16),
stabilityControlStatus	(17),
steeringWheelAngle	(18),
steeringWheelAngleRateOfChange	(19),
sunSensor	(20),
tankFilingLevel	(21),
temperature	(22),
tiltSensor	(23),
tirePressure	(24),
trafficDensity	(25),

```

    typeOfGoods          (26),
    vehicleElevation     (27),
    vehicleSize          (28),
    vehicleType          (29),
    visibilityRange      (30),
    weather              (31),
    windscreenWiper      (32),
    wiperRate            (33),
    yawRate              (34)
}

```

StdVANETStaticDataTagList ::= ENUMERATED {

/ Static elements sent by VANET without updating their value before the transmission. Data frames may be sent only without updating, therefore they belong to this set. If it's necessary to send the updated version of an element part of a frame, then that element must be included in the [StdVANETDynamicDataTagList](#).*/*

```

    brakeSystemStatus    (64),
    environmental         (65),
    RsuHmiMessage        (66),
    VehicleHmiMessage    (67),
    obstacleParameters    (68),
    systemsOnBoard       (69),
    trafficLightStatus    (70),
    trajectory            (71),
    vehicleMass           (72),
    reserved              (255)
}

```

SteeringWheelAngle ::= INTEGER (-32768..32767) -- LSB units of 0.02 degrees

*/*The angle of the steering wheel, expressed in a signed (to the right being positive) value with units of 0.02 degrees.*/*

SteeringWheelAngleRateOfChange ::= INTEGER (-128..127)

*/*The rate of change of the angle of the steering wheel, expressed in signed units of 3 degrees/second over a range of 381degrees in either direction. Being positive when steering wheel is turning clockwise.*/*

SunSensor ::= INTEGER (0..7) -- LSB units of 1

/ The "Sun Sensor" Probe Data Element is intended to inform Probe Data Users as to the level of Sun Light in the area the vehicle was traveling at the time the Probe Data snapshot was taken. The value of the Sun Sensor data element ranges from 0-7, with 0 indicating "Complete Darkness", 1 indicating "Minimal Sun Light", and 7 indicating "Maximum Sun Light". This information could be sent to vehicles approaching the area to tell drives to be prepared for sunny/clouding conditions ahead or a Weather Server for monitoring weather conditions in the area.*/*

TankFilingLevel ::= INTEGER (0..200) *-- LSB unit of 0.5%*

Temperature ::= INTEGER (0..191) *-- in deg C with a -40 offset*

TiltSensor ::= BOOLEAN *-- Activated = TRUE, Not Activated = FALSE*

TimeOn ::= INTEGER (0..255)

/ occupation time in steps of 0.01 sec., 100 is 1 second*/*

TirePressure ::= INTEGER (0..63) *-- LSB units of 0.1 bar*

TLCStatus ::= ENUMERATED {
 operating, (1),
 manual operation (policeman), (2),
 out of operation / technical failure, (3)
 }

TrafficDensity ::= SEQUENCE {

/ The traffic density is described in 3 bits. All bit set to '0' means unknown conditions. Currently are proposed 5 status but some bits are reserved for future system improvement */*

trafficHeading	Heading ,
trafficDensity	ENUMERATED {
unknown	(0),
regular	(1),
queuing / congested	(2),
slow traffic	(3),
heavy traffic	(4),
reserved	(8)
	}
	}

TravelDistance ::= INTEGER (1..8) *-- LSB units of 1*

TypeOfGoods ::= INTEGER (0..127) */*Type of goods on board defined according to ADR. 0 = data not available or not significant */*

TypeOfTrajectory ::= INTEGER (0..255)

/ The right is "type of manoeuvre" e.g. lane change, overtaking etc. */*

UseOfTelephone ::= ENUMERATED {

notAvailable (0), -- *Not available*
 off (1), -- *Off*
 on (2) -- *On*
 }

VanetMessageType ::= ENUMERATED {

CooperativeAwarenessMessage (0),
 beacon (1),
 awarenessCooperativeMessage (2),
 emergency (3),
 event (4),
 periodic (5),
 hmiEvent (6),
 hmiPeriodic (7),
 cvisBeacon (8),
 ...
 } -- 4 bits

VehClass ::= [VehicleDescription](#)

VehicleAttribute ::= BIT STRING {

stdSAFESPOT (0), -- B'000
 safeprobe (1), -- B'001
 crashed (2), -- B'010
 cvis (3), -- B'100

 reserved () -- B'111
 }

VehicleDescription ::= ENUMERATED {

/ tpeg description (alternative to previous SAE description – To be confirmed by C2C) */*

unknown (0),
 car (1),
 light goods vehicle (2),

heavy goods vehicle	(3),
public transport vehicle	(4),
pedal cycle	(5),
emergency vehicle	(6),
works vehicle	(7),
exceptional size vehicle	(8),
vehicle with trailer	(9),
high-sided vehicle	(10),
minibus	(11),
taxi	(12),
tram	(13),
trolley-bus	(14),
train	(15),
post bus	(16),
school bus	(17),
military vehicle	(18),
motorcycle	(19),
sledge	(20),
assistance vehicle	(21),
vehicle	(31)
}	

VehicleHeight ::= INTEGER (0..255) -- LSB units of 0.05 m

/ The height of the vehicle excluding any antenna(s), and expressed in units of 5 cm. In cases of vehicles with adjustable ride heights, camper shells, and other devices which may cause the overall height to vary, the largest possible height will be used.*/*

VehicleID ::= INTEGER (0..4294967295) -- LSB units of 1

VehicleLength ::= INTEGER (0..16383) -- LSB units of 0.01 m

/ The length of the vehicle expressed in centimeters, unsigned. Note that this is a 14 bit value and it is combined with a 10 bit value to form a 3 byte data frame. When sent alone it shall occupy 2 bytes with the upper two bits being set to zero.*/*

VehicleWidth ::= INTEGER (0..1023) -- LSB units of 0.01 m

/ The width of the vehicle expressed in centimeters, unsigned. Note that this is a 10 bit value and it is combined with a 14 bit value to form a 3 byte data frame. When sent alone it shall occupy 2 bytes with the upper six bits being set to zero.*/*

VerticalAcceleration ::= INTEGER (-127..127) -- LSB units of 0.08 m/s²
-- Actual range: +/- 1G range

VisibilityRange ::= INTEGER (0..255) -- LSB units of 5 m
 -- Actual range: 0 - 200 (0 - 1000 m.)
 -- 255 means data not available.

Weather ::= INTEGER (0..15) -- LSB units of 1
 /*TPEG : rain, fog, smoke, dust cloud, sand storm, spray, snow spray, blizzard (rtm17); sleet, hail (rtm29);*/

WidthLane ::= INTEGER (0..100) -- LSB units of 0.1 m

WindscreenWiper ::= ENUMERATED {
 notEquipped (0),
 off (1),
 intermittent (2),
 low (3),
 high (4),
 automaticPresent (255) -- Auto wiper equipped
 }

WiperRate ::= INTEGER (0..255) -- Units of sweeps per minute
 /* The current rate at which wiper sweeps are taking place on the subject vehicle. In units of sweeps per minute. Use a value of 1 for any sweep rate with a period greater than 60 seconds.*/