Commission Implementing Regulation (EU) 2016/799 of 18 March 2016 implementing Regulation (EU) No 165/2014 of the European Parliament and of the Council laying down the requirements for the construction, testing, installation, operation and repair of tachographs and their components (Text with EEA relevance)

ANNEX I C

Requirements for construction, testing, installation, and inspection

Appendix 1

DATA DICTIONARY

1. INTRODUCTION

This appendix specifies data formats, data elements, and data structures for use within the recording equipment and tachograph cards.

1.1. Approach for definitions of data types

This appendix uses Abstract Syntax Notation One (ASN.1) to define data types. This enables simple and structured data to be defined without implying any specific transfer syntax (encoding rules) which will be application and environment dependent.

ASN.1 type naming conventions are done in accordance with ISO/IEC 8824-1. This implies that:

— where possible, the meaning of the data type is implied through the names being selected,

— where a data type is a composition of other data types, the data type name is still a single sequence of alphabetical characters commencing with a capital letter, however capitals are used within the name to impart the corresponding meaning,

— in general, the data types names are related to the name of the data types from which they are constructed, the equipment in which data is stored and the function related to the data.

If an ASN.1 type is already defined as part of another standard and if it is relevant for usage in the recording equipment, then this ASN.1 type will be defined in this appendix.

To enable several types of encoding rules, some ASN.1 types in this appendix are constrained by value range identifiers. The value range identifiers are defined in paragraph 3 and Appendix 2.

1.2. References

The following references are used in this Appendix:

ISO 639	Code for the representation of names of languages. First Edition: 1988.
ISO 3166	Codes for the representation of names of countries and their subdivisions — Part 1: Country codes, 2013
ISO 3779	Road vehicles — Vehicle identification number (VIN) — Content and structure. 2009
ISO/IEC 7816-5	Identification cards — Integrated circuit cards — Part 5: Registration of application providers. Second edition: 2004.
ISO/IEC 7816-6	Identification cards — Integrated circuit cards — Part 6: Interindustry data elements for interchange, 2004 + Technical Corrigendum 1: 2006
ISO/IEC 8824-1	Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation. 2008 + Technical Corrigendum 1: 2012 and Technical Corrigendum 2: 2014.
ISO/IEC 8825-2	Information technology — ASN.1 encoding rules: Specification of Packed Encoding Rules (PER). 2008.
ISO/IEC 8859-1	Information technology — 8 bit single-byte coded graphic character sets — Part 1: Latin alphabet No.1. First edition: 1998.
ISO/IEC 8859-7	Information technology — 8 bit single-byte coded graphic character sets — Part 7: Latin/Greek alphabet. 2003.

ISO 16844-3	Road vehicles — Tachograph systems — Motion Sensor Interface. 2004 + Technical Corrigendum 1: 2006..
TR-03110-3	BSI / ANSSI Technical Guideline TR-03110-3, Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token — Part 3 Common Specifications, version 2.20, 3. February 2015

2. DATA TYPE DEFINITIONS

For any of the following data types, the default value for an ‘unknown’ or a ‘not applicable’ content will consist in filling the data element with ‘FF’ bytes.

All data types are used for Generation 1 and Generation 2 applications unless otherwise specified.

[[F1]For card data types used for Generation 1 and Generation 2 applications, the size specified in this Appendix is the one for Generation 2 application. The size for Generation 1 application is supposed to be already known by the reader. The Annex IC requirement numbers related to such data types cover both Generation 1 and Generation 2 applications.**]**

Textual Amendments

F1 Inserted by Commission Implementing Regulation (EU) 2018/502 of 28 February 2018 amending Implementing Regulation (EU) 2016/799 laying down the requirements for the construction, testing, installation, operation and repair of tachographs and their components (Text with EEA relevance).

2.1. **ActivityChangeInfo**

This data type enables to code, within a two bytes word, a slot status at 00:00 and/or a driver status at 00:00 and/or changes of activity and/or changes of driving status and/or changes of card status for a driver or a co-driver. This data type is related to Annex 1C requirements 105, 266, 291, 320, 321, 343, and 344.

```
ActivityChangeInfo ::= OCTET STRING (SIZE(2))
```

Value assignment — Octet Aligned: ‘scpaatttttttttt’B (16 bits)

For Data Memory recordings (or slot status):

‘s’B	Slot: ‘0’B: DRIVER, ‘1’B: CO-DRIVER,
‘c’B	Driving status: ‘0’B: SINGLE, ‘1’B: CREW,
‘p’B	Driver (or workshop) card status in the relevant slot: ‘0’B: INSERTED, a card is inserted, ‘1’B: NOT INSERTED, no card is inserted (or a card is withdrawn),
‘aa’B	Activity: ‘00’B: BREAK/REST, ‘01’B: AVAILABILITY, ‘10’B: WORK, ‘11’B: DRIVING,
‘ttttttttttt’B	Time of the change: Number of minutes since 00h00 on the given day.

For Driver (or Workshop) card recordings (and driver status):

‘s’B	Slot (not relevant when ‘p’=1 except note below): ‘0’B: DRIVER, ‘1’B: CO-DRIVER,
‘c’B	Driving status (case ‘p’=0) or Following activity status (case ‘p’=1): ‘0’B: SINGLE, ‘0’B: UNKNOWN ‘1’B: CREW, ‘1’B: KNOWN (=manually entered)
‘p’B	Card status: ‘0’B: INSERTED, the card is inserted in a recording equipment, ‘1’B: NOT INSERTED, the card is not inserted (or the card is withdrawn),
‘aa’B	Activity (not relevant when ‘p’=1 and ‘c’=0 except note below): ‘00’B: BREAK/REST, ‘01’B: AVAILABILITY, ‘10’B: WORK, ‘11’B: DRIVING,
‘ttttttttttt’B	Time of the change: Number of minutes since 00h00 on the given day.

Note for the case ‘card withdrawal’:

When the card is withdrawn:

— ‘s’ is relevant and indicates the slot from which the card is withdrawn,

— ‘c’ must be set to 0,

— ‘p’ must be set to 1,

— ‘aa’ must code the current activity selected at that time,

As a result of a manual entry, the bits ‘c’ and ‘aa’ of the word (stored in a card) may be overwritten later to reflect the entry.

2.2. **Address**

An address.

```
Address ::= SEQUENCE {
   codePage                                INTEGER (0..255),
   address                                 OCTET STRING (SIZE(35))
}
```

codePage specifies a character set defined in Chapter 4,

address is an address encoded using the specified character set.

2.3. **AESKey**

Generation 2:

An AES key with a length of 128, 192 or 256 bits.

```
AESKey ::= CHOICE {
   aes128Key                                   AES128Key,
   aes192Key                                   AES192Key,
   aes256Key                                   AES256Key
}
```

Value assignment: not further specified.

2.4. **AES128Key**

Generation 2:

An AES128 key.

```
AES128Key ::= SEQUENCE {
   length                                      INTEGER(0..255),
   aes128Key                                   OCTET STRING (SIZE(16))
}
```

length denotes the length of the AES128 key in octets.

aes128Key is an AES key with a length of 128 bits.

Value assignment:

The length shall have the value 16.

2.5. **AES192Key**

Generation 2:

An AES192 key.

```
AES192Key ::= SEQUENCE {
   length                                      INTEGER(0..255),
   aes192Key                                   OCTET STRING (SIZE(24))
}
```

length denotes the length of the AES192 key in octets.

aes192Key is an AES key with a length of 192 bits.

Value assignment:

The length shall have the value 24.

2.6. **AES256Key**

Generation 2:

An AES256 key.

```
AES256Key ::= SEQUENCE {
   length                                      INTEGER(0..255),
   aes256Key                                   OCTET STRING (SIZE(32))
}
```

length denotes the length of the AES256 key in octets.

aes256Key is an AES key with a length of 256 bits.

Value assignment:

The length shall have the value 32.

2.7. BCDString

BCDString is applied for Binary Code Decimal (BCD) representation. This data type is used to represent one decimal digit in one semi octet (4 bits). BCDString is based on the ISO/IEC 8824-1 'CharacterStringType'.

```
BCDString ::= CHARACTER STRING (WITH COMPONENTS {
   identification ( WITH COMPONENTS {
   fixed PRESENT }) })
```

BCDString uses an 'hstring' notation. The leftmost hexadecimal digit shall be the most significant semi octet of the first octet. To produce a multiple of octets, zero trailing semi octets shall be inserted, as needed, from the leftmost semi octet position in the first octet.

Permitted digits are: 0, 1, .. 9.

2.8. CalibrationPurpose

Code explaining why a set of calibration parameters was recorded. This data type is related to Annex 1B requirements 097 and 098 and Annex 1C requirements 119.

```
CalibrationPurpose ::= OCTET STRING (SIZE(1))
```

Value assignment:

Generation 1:

'00'H	reserved value,
'01'H	activation: recording of calibration parameters known, at the moment of the VU activation,
'02'H	first installation: first calibration of the VU after its activation,
'03'H	installation: first calibration of the VU in the current vehicle,
'04'H	periodic inspection.

Generation 2:

In addition to generation 1 the following values are used:

'05'H	entry of VRN by company,
'06'H	time adjustment without calibration,
'07'H to '7F'H	RFU,
'80'H to 'FF'H	Manufacturer specific.

2.9. CardActivityDailyRecord

Information, stored in a card, related to the driver activities for a particular calendar day. This data type is related to Annex 1C requirements 266, 291, 320 and 343.

```
CardActivityDailyRecord ::= SEQUENCE {
   activityPreviousRecordLength         INTEGER(0..CardActivityLengthRange),
   activityRecordLength                 INTEGER(0..CardActivityLengthRange),
   activityRecordDate                   TimeReal,
   activityDailyPresenceCounter         DailyPresenceCounter,
   activityDayDistance                  Distance,
   activityChangeInfo                   SET SIZE(1..1440) OF ActivityChangeInfo
}
```

activityPreviousRecordLength is the total length in bytes of the previous daily record. The maximum value is given by the length of the OCTET STRING containing these records (see CardActivityLengthRange Appendix 2 paragraph 4). When this record is the oldest daily record, the value of activityPreviousRecordLength must be set to 0.

activityRecordLength is the total length in bytes of this record. The maximum value is given by the length of the OCTET STRING containing these records.

activityRecordDate is the date of the record.

activityDailyPresenceCounter is the daily presence counter for the card this day.

activityDayDistance is the total distance travelled this day.

activityChangeInfo is the set of ActivityChangeInfo data for the driver this day. It may contain at maximum 1440 values (one activity change per minute). This set always includes the activityChangeInfo coding the driver status at 00:00.

2.10. CardActivityLengthRange

Number of bytes in a driver or a workshop card, available to store driver activity records.

```
CardActivityLengthRange ::= INTEGER(0..2^16-1)
```

Value assignment: see Appendix 2.

2.11. CardApprovalNumber

Type approval number of the card.

```
CardApprovalNumber ::= IA5String(SIZE(8))
```

Value assignment:

The approval number shall be provided as published on the corresponding European Commission web site, i.e. for example including hyphens if any. The approval number shall be left-aligned.

2.12. CardCertificate

Generation 1:

Certificate of the public key of a card.

```
CardCertificate ::= Certificate
```

2.13. CardChipIdentification

Information, stored in a card, related to the identification of the card's Integrated Circuit (IC) (Annex 1C requirement 249). The icSerialNumber together with the icManufacturingReferences identifies the card chip uniquely. The icSerialNumber alone does not uniquely identify the card chip.

```
CardChipIdentification ::= SEQUENCE {
   icSerialNumber                    OCTET STRING (SIZE(4)),
   icManufacturingReferences         OCTET STRING (SIZE(4))
}
```

icSerialNumber is the IC serial number.

icManufacturingReferences is the IC manufacturer specific identifier.

2.14. **CardConsecutiveIndex**

A card consecutive index (definition h)).

```
CardConsecutiveIndex ::= IA5String(SIZE(1))
```

Value assignment: (see Annex 1C chapter 7)

Order for increase: '0, …, 9, A, …, Z, a, …, z'

2.15. **CardControlActivityDataRecord**

Information, stored in a driver or workshop card, related to the last control the driver has been subject to (Annex 1C requirements 274, 299, 327, and 350).

```
CardControlActivityDataRecord ::= SEQUENCE {
   controlType                       ControlType,
   controlTime                       TimeReal,
   controlCardNumber                 FullCardNumber,
   controlVehicleRegistration        VehicleRegistrationIdentification,
   controlDownloadPeriodBegin        TimeReal,
   controlDownloadPeriodEnd          TimeReal
}
```

controlType is the type of the control.

controlTime is the date and time of the control.

controlCardNumber is the FullCardNumber of the control officer having performed the control.

controlVehicleRegistration is the VRN and registering Member State of the vehicle in which the control happened.

controlDownloadPeriodBegin and **controlDownloadPeriodEnd** is the period downloaded, in case of downloading.

2.16. **CardCurrentUse**

Information about the actual usage of the card (Annex 1C requirement 273, 298, 326, and 349).

```
CardCurrentUse ::= SEQUENCE {
   sessionOpenTime                   TimeReal,
   sessionOpenVehicle                VehicleRegistrationIdentification
}
```

sessionOpenTime is the time when the card is inserted for the current usage. This element is set to zero at card removal.

sessionOpenVehicle is the identification of the currently used vehicle, set at card insertion. This element is set to zero at card removal.

2.17. **CardDriverActivity**

Information, stored in a driver or a workshop card, related to the activities of the driver (Annex 1C requirements 267, 268, 292, 293, 321 and 344).

```
CardDriverActivity ::= SEQUENCE {
   activityPointerOldestDayRecord       INTEGER(0.. CardActivityLengthRange-1),
   activityPointerNewestRecord          INTEGER(0.. CardActivityLengthRange-1),
   activityDailyRecords                 OCTET STRING
                                        (SIZE(CardActivityLengthRange))
}
```

activityPointerOldestDayRecord is the specification of the begin of the storage place (number of bytes from the beginning of the string) of the oldest complete day record in the activityDailyRecords string. The maximum value is given by the length of the string.

activityPointerNewestRecord is the specification of the begin of the storage place (number of bytes from the beginning of the string) of the most recent day record in the activityDailyRecords string. The maximum value is given by the length of the string.

activityDailyRecords is the space available to store the driver activity data (data structure: CardActivityDailyRecord) for each calendar day where the card has been used.

Value assignment: this octet string is cyclically filled with records of CardActivityDailyRecord. At the first use storing is started at the first byte of the string. All new records are appended at the end of the previous one. When the string is full, storing continues at the first byte of the string independently of a break being inside a data element. Before placing new activity data in the string (enlarging current activityDailyRecord, or placing a new activityDailyRecord) that replaces older activity data, activityPointerOldestDayRecord must be updated to reflect the new location of the oldest complete day record, and activityPreviousRecordLength of this (new) oldest complete day record must be reset to 0.

2.18. CardDrivingLicenceInformation

Information, stored in a driver card, related to the card holder driver licence data (Annex 1C requirement 259 and 284).

```
CardDrivingLicenceInformation ::= SEQUENCE {
   drivingLicenceIssuingAuthority       Name,
   drivingLicenceIssuingNation          NationNumeric,
   drivingLicenceNumber                 IA5String(SIZE(16))
}
```

drivingLicenceIssuingAuthority is the authority responsible for issuing the driving licence.

drivingLicenceIssuingNation is the nationality of the authority that issued the driving licence.

drivingLicenceNumber is the number of the driving licence.

[F2 2.19. CardEventData

Generation 1:

Information, stored in a driver or workshop card, related to the events associated with the card holder (Annex IC requirements 260 and 318).

```
CardEventData ::= SEQUENCE SIZE(6) OF {
  cardEventRecords                    SET SIZE(NoOfEventsPerType) OF
                                      CardEventRecord
}
```

CardEventData is a sequence, ordered by ascending value of EventFaultType, of cardEventRecords (except security breach attempts related records which are gathered in the last set of the sequence).

cardEventRecords is a set of event records of a given event type (or category for security breach attempts events).

Generation 2:

Information, stored in a driver or workshop card, related to the events associated with the card holder (Annex IC requirements 285 and 341).

```
CardEventData ::= SEQUENCE SIZE(11) OF {
   cardEventRecords                    SET SIZE(NoOfEventsPerType) OF
                                       CardEventRecord
}
```

CardEventData is a sequence, ordered by ascending value of EventFaultType, of cardEventRecords (except security breach attempts related records which are gathered in the last set of the sequence).

cardEventRecords is a set of event records of a given event type (or category for security breach attempts events).**]**

Textual Amendments

F2 Substituted by Commission Implementing Regulation (EU) 2018/502 of 28 February 2018 amending Implementing Regulation (EU) 2016/799 laying down the requirements for the construction, testing, installation, operation and repair of tachographs and their components (Text with EEA relevance).

2.20. CardEventRecord

Information, stored in a driver or a workshop card, related to an event associated to the card holder (Annex 1C requirements 261, 286, 318 and 341).

```
CardEventRecord ::= SEQUENCE {
   eventType                           EventFaultType,
   eventBeginTime                      TimeReal,
   eventEndTime                        TimeReal,
   eventVehicleRegistration            VehicleRegistrationIdentification
}
```

eventType is the type of the event.

eventBeginTime is the date and time of beginning of event.

eventEndTime is the date and time of end of event.

eventVehicleRegistration is the VRN and registering Member State of vehicle in which the event happened.

2.21. CardFaultData

Information, stored in a driver or a workshop card, related to the faults associated to the card holder (Annex 1C requirements 263, 288, 318, and 341).

```
CardFaultData ::= SEQUENCE SIZE(2) OF {
   cardFaultRecords                    SET SIZE(NoOfFaultsPerType) OF
                                       CardFaultRecord
}
```

CardFaultData is a sequence of Recording Equipment faults set of records followed by card faults set of records.

cardFaultRecords is a set of fault records of a given fault category (Recording Equipment or card).

2.22. CardFaultRecord

Information, stored in a driver or a workshop card, related to a fault associated to the card holder (Annex 1C requirement 264, 289, 318, and 341).

```
CardFaultRecord ::= SEQUENCE {
   faultType                           EventFaultType,
   faultBeginTime                      TimeReal,
   faultEndTime                        TimeReal,
   faultVehicleRegistration            VehicleRegistrationIdentification
}
```

faultType is the type of the fault.

faultBeginTime is the date and time of beginning of fault.

faultEndTime is the date and time of end of fault.

faultVehicleRegistration is the VRN and registering Member State of vehicle in which the fault happened.

2.23. CardIccIdentification

Information, stored in a card, related to the identification of the integrated circuit (IC) card (Annex 1C requirement 248).

```
CardIccIdentification ::= SEQUENCE {
   clockStop                           OCTET STRING (SIZE(1)),
   cardExtendedSerialNumber            ExtendedSerialNumber,
   cardApprovalNumber                  CardApprovalNumber,
   cardPersonaliserID                  ManufacturerCode,
   embedderIcAssemblerId               EmbedderIcAssemblerId,
   icIdentifier                        OCTET STRING (SIZE(2))
}
```

clockStop is the Clockstop mode as defined in appendix 2.

cardExtendedSerialNumber is the IC card unique serial number as further specified by the ExtendedSerialNumber data type.

cardApprovalNumber is the type approval number of the card.

cardPersonaliserID is the card personaliser ID encoded as ManufacturerCode.

embedderIcAssemblerId provides information about the embedder/IC assembler.

icIdentifier is the Identifier of the IC on the card and its IC manufacturer as defined in ISO/IEC 7816-6.

2.24. CardIdentification

Information, stored in a card, related to the identification of the card (Annex 1C requirements 255, 280, 310, 333, 359, 365, 371, and 377).

```
CardIdentification ::= SEQUENCE {
   cardIssuingMemberState                NationNumeric,
   cardNumber                            CardNumber,
   cardIssuingAuthorityName              Name,
   cardIssueDate                         TimeReal,
   cardValidityBegin                     TimeReal,
   cardExpiryDate                        TimeReal
}
```

cardIssuingMemberState is the code of the Member State issuing the card.

cardNumber is the card number of the card.

cardIssuingAuthorityName is the name of the authority having issued the Card.

cardIssueDate is the issue date of the Card to the current holder.

cardValidityBegin is the first date of validity of the card.

cardExpiryDate is the date when the validity of the card ends.

2.25. CardMACertificate

Generation 2:

Certificate of the card public key for mutual authentication with a VU. The structure of this certificate is specified in Appendix 11.

```
CardMACertificate ::= Certificate
```

2.26. CardNumber

A card number as defined by definition g).

```
CardNumber ::= CHOICE {
   SEQUENCE {
      driverIdentification               IA5String(SIZE(14)),
      cardReplacementIndex               CardReplacementIndex,
      cardRenewalIndex                   CardRenewalIndex
   },
   SEQUENCE {
      ownerIdentification                IA5String(SIZE(13)),
      cardConsecutiveIndex               CardConsecutiveIndex,
      cardReplacementIndex               CardReplacementIndex,
      cardRenewalIndex                   CardRenewalIndex
   }
}
```

driverIdentification is the unique identification of a driver in a Member State.

ownerIdentification is the unique identification of a company or a workshop or a control body within a member state.

cardConsecutiveIndex is the card consecutive index.

cardReplacementIndex is the card replacement index.

cardRenewalIndex is the card renewal index.

The first sequence of the choice is suitable to code a driver card number, the second sequence of the choice is suitable to code workshop, control, and company card numbers.

2.27. **CardPlaceDailyWorkPeriod**

Information, stored in a driver or a workshop card, related to the places where daily work periods begin and/or end (Annex 1C requirements 272, 297, 325, and 348).

```
CardPlaceDailyWorkPeriod ::= SEQUENCE {
   placePointerNewestRecord    INTEGER(0 .. NoOfCardPlaceRecords-1),
   placeRecords                SET SIZE(NoOfCardPlaceRecords) OF PlaceRecord
}
```

placePointerNewestRecord is the index of the last updated place record.

Value assignment: Number corresponding to the numerator of the place record, beginning with '0' for the first occurrence of the place records in the structure.

placeRecords is the set of records containing the information related to the places entered.

2.28. **CardPrivateKey**

Generation 1:

The private key of a card.

```
CardPrivateKey ::= RSAKeyPrivateExponent
```

2.29. **CardPublicKey**

The public key of a card.

```
CardPublicKey ::= PublicKey
```

[[F2]2.30. **CardRenewalIndex**

A card renewal index (definition i)).

```
CardRenewalIndex::= IA5String(SIZE(1))
```

Value assignment: (see this Annex chapter 7).

'0' First issue.

Order for increase: '0, …, 9, A, …, Z'**]**

2.31. **CardReplacementIndex**

A card replacement index (definition j)).

```
CardReplacementIndex ::= IA5String(SIZE(1))
```

Value assignment: (see this Annex chapter VII).

'0' Original card.

Order for increase: '0, …, 9, A, …, Z'

2.32. **CardSignCertificate**

Generation 2:

Certificate of the card public key for signature. The structure of this certificate is specified in Appendix 11.

```
CardSignCertificate ::= Certificate
```

2.33. **CardSlotNumber**

Code to distinguish between the two slots of a Vehicle Unit.

```
CardSlotNumber ::= INTEGER {
   driverSlot                              (0),
   co-driverSlot                           (1)
}
```

Value assignment: not further specified.

2.34. **CardSlotsStatus**

Code indicating the type of cards inserted in the two slots of the vehicle unit.

```
CardSlotsStatus ::= OCTET STRING (SIZE(1))
```

Value assignment — Octet Aligned: ‘ccccdddd’B

‘cccc’B	Identification of the type of card inserted in the co-driver slot,
‘dddd’B	Identification of the type of card inserted in the driver slot,

with the following identification codes:

‘0000’B	no card is inserted,
‘0001’B	a driver card is inserted,
‘0010’B	a workshop card is inserted,
‘0011’B	a control card is inserted,
‘0100’B	a company card is inserted.

2.35. **CardSlotsStatusRecordArray**

Generation 2:

The CardSlotsStatus plus metadata as used in the download protocol.

```
CardSlotsStatusRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF CardSlotsStatus
}
```

recordType denotes the type of the record (CardSlotsStatus). **Value Assignment**: See RecordType

recordSize is the size of the CardSlotsStatus in bytes.

noOfRecords is the number of records in the set records.

records is the set of CardSlotsStatus records.

2.36. **CardStructureVersion**

Code indicating the version of the implemented structure in a tachograph card.

```
CardStructureVersion ::= OCTET STRING (SIZE(2))
```

Value assignment: ‘aabb’H:

'aa'H	Index for changes of the structure. '00'H for Generation 1 applications '01'H for Generation 2 applications
'bb'H	Index for changes concerning the use of the data elements defined for the structure given by the high byte. '00'H for this version of Generation 1 applications '00'H for this version of Generation 2 applications

2.37. **CardVehicleRecord**

Information, stored in a driver or workshop card, related to a period of use of a vehicle during a calendar day (Annex 1C requirements 269, 294, 322, and 345).

Generation 1:

vehicleOdometerBegin is the vehicle odometer value at the beginning of the period of use of the vehicle.

vehicleOdometerEnd is the vehicle odometer value at the end of the period of use of the vehicle.

vehicleFirstUse is the date and time of the beginning of the period of use of the vehicle.

vehicleLastUse is the date and time of the end of the period of use of the vehicle.

vehicleRegistration is the VRN and the registering Member State of the vehicle.

vuDataBlockCounter is the value of the VuDataBlockCounter at last extraction of the period of use of the vehicle.

Generation 2:

In addition to generation 1 the following data element is used:

VehicleIdentificationNumber is the vehicle identification number referring to the vehicle as a whole.

2.38. **CardVehiclesUsed**

Information, stored in a driver or workshop card, related to the vehicles used by the card holder (Annex 1C requirements 270, 295, 323, and 346).

```
CardVehiclesUsed := SEQUENCE {
   vehiclePointerNewestRecord          INTEGER(0..NoOfCardVehicleRecords-1),
   cardVehicleRecords                  SET SIZE(NoOfCardVehicleRecords) OF
                                       CardVehicleRecord
}
```

vehiclePointerNewestRecord is the index of the last updated vehicle record.

Value assignment: Number corresponding to the numerator of the vehicle record, beginning with '0' for the first occurrence of the vehicle records in the structure.

cardVehicleRecords is the set of records containing information on vehicles used.

2.39. **CardVehicleUnitRecord**

Generation 2:

Information, stored in a driver or workshop card, related to a vehicle unit that was used (Annex 1C requirement 303 and 351).

```
CardVehicleUnitRecord ::= SEQUENCE {
   timeStamp                                TimeReal,
   manufacturerCode                         ManufacturerCode,
   deviceID                                 INTEGER(0..255),
   vuSoftwareVersion                        VuSoftwareVersion
}
```

timeStamp is the beginning of the period of use of the vehicle unit (i.e. first card insertion in the vehicle unit for the period).

manufacturerCode identifies the manufacturer of the Vehicle Unit.

deviceID identifies the Vehicle Unit type of a manufacturer. The value is manufacturer specific.

vuSoftwareVersion is the software version number of the Vehicle Unit.

2.40. **CardVehicleUnitsUsed**

Generation 2:

Information, stored in a driver or workshop card, related to the vehicle units used by the card holder (Annex 1C requirement 306 and 352).

```
CardVehicleUnitsUsed := SEQUENCE {
   vehicleUnitPointerNewestRecord       INTEGER(0..NoOfCardVehicleUnitRecords-1),
   cardVehicleUnitRecords               SET SIZE(NoOfCardVehicleUnitRecords) OF
                                        CardVehicleUnitRecord
}
```

vehicleUnitPointerNewestRecord is the index of the last updated vehicle unit record.

Value assignment: Number corresponding to the numerator of the vehicle unit record, beginning with '0' for the first occurrence of the vehicle unit records in the structure.

cardVehicleUnitRecords is the set of records containing information on vehicle units used.

2.41. **Certificate**

The certificate of a public key issued by a Certification Authority.

Generation 1:

Value assignment: digital signature with partial recovery of a CertificateContent according to Appendix 11 common security mechanisms: Signature (128 bytes) || Public Key remainder (58 bytes) || Certification Authority Reference (8 bytes).

Generation 2:

Value assignment: See Appendix 11

2.42. **CertificateContent**

Generation 1:

The (clear) content of the certificate of a public key according to Appendix 11 common security mechanisms.

```
CertificateContent ::= SEQUENCE {
   certificateProfileIdentifier          INTEGER(0..255),
   certificationAuthorityReference       KeyIdentifier,
   certificateHolderAuthorisation        CertificateHolderAuthorisation,
   certificateEndOfValidity              TimeReal,
   certificateHolderReference            KeyIdentifier,
   publicKey                             PublicKey
}
```

certificateProfileIdentifier is the version of the corresponding certificate.

Value assignment: '01h' for this version.

certificationAuthorityReference identifies the Certification Authority issuing the certificate. It also references the Public Key of this Certification Authority.

certificateHolderAuthorisation identifies the rights of the certificate holder.

certificateEndOfValidity is the date when the certificate expires administratively.

certificateHolderReference identifies the certificate holder. It also references his Public Key.

publicKey is the public key that is certified by this certificate.

2.43. **CertificateHolderAuthorisation**

Identification of the rights of a certificate holder.

```
CertificateHolderAuthorisation ::= SEQUENCE {
   tachographApplicationID               OCTET STRING(SIZE(6))
   equipmentType                         EquipmentType
}
```

Generation 1:

tachographApplicationID is the application identifier for the tachograph application.

Value assignment: 'FFh''54h''41h''43h''48h''4Fh'. This AID is a proprietary non registered application identifier in accordance with ISO/IEC 7816-5.

equipmentType is the identification of the type of equipment to which the certificate is intended.

Value assignment: in accordance with EquipmentType data type. **0** if certificate is the one of a Member State.

Generation 2:

tachographApplicationID denotes the 6 most significant bytes of the generation 2 tachograph card application identifier (AID). The AID for the tachograph card application is specified in chapter 6.2.

Value assignment:'FF 53 4D 52 44 54'.

equipmentType is the identification of the type of equipment as specified for generation 2 to which the certificate is intended.

Value assignment: in accordance with EquipmentType data type.

2.44. **CertificateRequestID**

Unique identification of a certificate request. It can also be used as a Vehicle Unit Public Key Identifier if the serial number of the vehicle Unit to which the key is intended is not known at certificate generation time.

```
CertificateRequestID ::= SEQUENCE{
   requestSerialNumber                    INTEGER(0..2^32-1),
   requestMonthYear                       BCDString(SIZE(2)),
   crIdentifier                           OCTET STRING(SIZE(1)),
   manufacturerCode                       ManufacturerCode
}
```

requestSerialNumber is a serial number for the certificate request, unique for the manufacturer and the month below.

requestMonthYear is the identification of the month and the year of the certificate request.

Value assignment: BCD coding of Month (two digits) and Year (two last digits).

crIdentifier: is an identifier to distinguish a certificate request from an extended serial number.

Value assignment: ‘FFh’.

manufacturerCode: is the numerical code of the manufacturer requesting the certificate.

2.45. **CertificationAuthorityKID**

Identifier of the Public Key of a Certification Authority (a Member State or the European Certification Authority).

```
CertificationAuthorityKID ::= SEQUENCE{
   nationNumeric                          NationNumeric,
   nationAlpha                            NationAlpha,
   keySerialNumber                        INTEGER(0..255),
   additionalInfo                         OCTET STRING(SIZE(2)),
   caIdentifier                           OCTET STRING(SIZE(1))
}
```

nationNumeric is the numerical nation code of the Certification Authority.

nationAlpha is the alphanumerical nation code of the Certification Authority.

keySerialNumber is a serial number to distinguish the different keys of the Certification Authority in the case keys are changed.

additionalInfo is a two byte field for additional coding (Certification Authority specific).

caIdentifier is an identifier to distinguish a Certification Authority Key Identifier from other Key Identifiers.

Value assignment: ‘01h’.

2.46. **CompanyActivityData**

Information, stored in a company card, related to activities performed with the card (Annex 1C requirement 373 and 379).

```
CompanyActivityData ::= SEQUENCE {
   companyPointerNewestRecord        INTEGER(0..NoOfCompanyActivityRecords-1),
   companyActivityRecords            SET SIZE(NoOfCompanyActivityRecords) OF
    companyActivityRecord             SEQUENCE {
     companyActivityType               CompanyActivityType,
     companyActivityTime               TimeReal,
     cardNumberInformation             FullCardNumber,
     vehicleRegistrationInformation    VehicleRegistrationIdentification,
     downloadPeriodBegin               TimeReal,
     downloadPeriodEnd                 TimeReal
    }
}
```

companyPointerNewestRecord is the index of the last updated companyActivityRecord.

Value assignment: Number corresponding to the numerator of the company activity record, beginning with '0' for the first occurrence of the company activity record in the structure.

companyActivityRecords is the set of all company activity records.

companyActivityRecord is the sequence of information related to one company activity.

companyActivityType is the type of the company activity.

companyActivityTime is the date and time of the company activity.

cardNumberInformation is the card number and the card issuing Member State of the card downloaded, if any.

vehicleRegistrationInformation is the VRN and registering Member State of the vehicle downloaded or locked in or out.

downloadPeriodBegin and **downloadPeriodEnd** is the period downloaded from the VU, if any.

2.47. CompanyActivityType

Code indicating an activity carried out by a company using its company card.

```
CompanyActivityType ::= INTEGER {
   card downloading                (1),
   VU downloading                  (2),
   VU lock-in                      (3),
   VU lock-out                     (4)
}
```

2.48. CompanyCardApplicationIdentification

Information, stored in a company card related to the identification of the application of the card (Annex 1C requirement 369 and 375).

```
CompanyCardApplicationIdentification ::= SEQUENCE {
   typeOfTachographCardId              EquipmentType,
   cardStructureVersion                CardStructureVersion,
   noOfCompanyActivityRecords          NoOfCompanyActivityRecords
}
```

typeOfTachographCardId is specifying the implemented type of card.

cardStructureVersion is specifying the the version of the structure that is implemented in the card.

noOfCompanyActivityRecords is the number of company activity records the card can store.

2.49. CompanyCardHolderIdentification

Information, stored in a company card, related to the cardholder identification (Annex 1C requirement 372 and 378).

```
CompanyCardHolderIdentification ::= SEQUENCE {
   companyName                          Name,
   companyAddress                       Address,
   cardHolderPreferredLanguage          Language
}
```

companyName is the name of the holder company.

companyAddress is the address of the holder company.

cardHolderPreferredLanguage is the preferred language of the card holder.

2.50. ControlCardApplicationIdentification

Information, stored in a control card related to the identification of the application of the card (Annex 1C requirement 357 and 363).

```
ControlCardApplicationIdentification ::= SEQUENCE {
   typeOfTachographCardId               EquipmentType,
   cardStructureVersion                 CardStructureVersion,
   noOfControlActivityRecords           NoOfControlActivityRecords
}
```

typeOfTachographCardId is specifying the implemented type of card.

cardStructureVersion is specifying the version of the structure that is implemented in the card.

noOfControlActivityRecords is the number of control activity records the card can store.

2.51. ControlCardControlActivityData

Information, stored in a control card, related to control activity performed with the card (Annex 1C requirement 361 and 367).

```
ControlCardControlActivityData ::= SEQUENCE {
   controlPointerNewestRecord        INTEGER(0.. NoOfControlActivityRecords-1),
   controlActivityRecords            SET SIZE(NoOfControlActivityRecords) OF
    controlActivityRecord             SEQUENCE {
     controlType                       ControlType,
     controlTime                       TimeReal,
     controlledCardNumber              FullCardNumber,
     controlledVehicleRegistration     VehicleRegistrationIdentification,
     controlDownloadPeriodBegin        TimeReal,
     controlDownloadPeriodEnd          TimeReal
    }
}
```

controlPointerNewestRecord is the index of the last updated control activity record.

Value assignment: Number corresponding to the numerator of the control activity record, beginning with '0' for the first occurrence of the control activity record in the structure.

controlActivityRecords is the set of all control activity records.

controlActivityRecord is the sequence of information related to one control.

controlType is the type of the control.

controlTime is the date and time of the control.

controlledCardNumber is the card number and the card issuing Member State of the card controlled.

controlledVehicleRegistration is the VRN and registering Member State of the vehicle in which the control happened.

controlDownloadPeriodBegin and **controlDownloadPeriodEnd** is the period eventually downloaded.

2.52. ControlCardHolderIdentification

Information, stored in a control card, related to the identification of the cardholder (Annex 1C requirement 360 and 366).

```
ControlCardHolderIdentification ::= SEQUENCE {
   controlBodyName                  Name,
   controlBodyAddress               Address,
   cardHolderName                   HolderName,
   cardHolderPreferredLanguage      Language
}
```

controlBodyName is the name of the control body of the card holder.

controlBodyAddress is the address of the control body of the card holder.

cardHolderName is the name and first name(s) of the holder of the Control Card.

cardHolderPreferredLanguage is the preferred language of the card holder.

2.53. ControlType

Code indicating the activities carried out during a control. This data type is related to Annex 1C requirements 126, 274, 299, 327, and 350.

```
ControlType ::= OCTET STRING (SIZE(1))
```

Generation 1:

Value assignment — Octet aligned: ‘cvpdxxxx’B (8 bits)

‘c’B card downloading:
- ‘0’B: card not downloaded during this control activity,
- ‘1’B: card downloaded during this control activity

‘v’B VU downloading:
- ‘0’B: VU not downloaded during this control activity,
- ‘1’B: VU downloaded during this control activity

‘p’B printing:
- ‘0’B: no printing done during this control activity,
- ‘1’B: printing done during this control activity

‘d’B	display: ‘0’B: no display used during this control activity, ‘1’B: display used during this control activity
‘xxxx’B	Not used.

Generation 2:

Value assignment — Octet aligned: ‘cvpdexxx’B (8 bits)

‘c’B	card downloading: ‘0’B: card not downloaded during this control activity, ‘1’B: card downloaded during this control activity
‘v’B	VU downloading: ‘0’B: VU not downloaded during this control activity, ‘1’B: VU downloaded during this control activity
‘p’B	printing: ‘0’B: no printing done during this control activity, ‘1’B: printing done during this control activity
‘d’B	display: ‘0’B: no display used during this control activity, ‘1’B: display used during this control activity
‘e’B	roadside calibration checking: ‘0’B: calibration parameters not checked during this control activity, ‘1’B: calibration parameters checked during this control activity
‘xxx’B	RFU.

2.54. CurrentDateTime

The current date and time of the recording equipment.

```
CurrentDateTime ::= TimeReal
```

Value assignment: not further specified.

2.55. CurrentDateTimeRecordArray

Generation 2:

The current date and time plus metadata as used in the download protocol.

```
CurrentDateTimeRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF CurrentDateTime
}
```

recordType denotes the type of the record (CurrentDateTime). **Value Assignment**: See RecordType

recordSize is the size of the CurrentDateTime in bytes.

noOfRecords is the number of records in the set records.

records is a set of current date and time records.

2.56. **DailyPresenceCounter**

Counter, stored in a driver or workshop card, increased by one for each calendar day the card has been inserted in a VU. This data type is related to Annex 1C requirements 266, 299, 320, and 343.

```
DailyPresenceCounter ::= BCDString(SIZE(2))
```

Value assignment: Consecutive Number with maximum value = 9 999, starting again with 0. At the time of first issuing of the card the number is set to 0.

2.57. **Datef**

Date expressed in a readily printable numeric format.

```
Datef ::= SEQUENCE {
   year         BCDString(SIZE(2)),
   month        BCDString(SIZE(1)),
   day          BCDString(SIZE(1))
}
```

Value assignment:

yyyy	Year
mm	Month
dd	Day
‘00000000’H	denotes explicitly no date.

2.58. **DateOfDayDownloaded**

Generation 2:

The date and time of the download.

```
DateOfDayDownloaded ::= TimeReal
```

Value assignment: not further specified.

2.59. **DateOfDayDownloadedRecordArray**

Generation 2:

The date and time of the download plus metadata as used in the download protocol.

```
DateofDayDownloadedRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       DateOfDayDownloaded
}
```

recordType denotes the type of the record (DateOfDayDownloaded). **Value Assignment**: See RecordType

recordSize is the size of the CurrentDateTime in bytes.

noOfRecords is the number of records in the set records.

records is the set of date and time of the download records.

2.60. **Distance**

A distance travelled (result of the calculation of the difference between two vehicle's odometer values in kilometers).

```
Distance ::= INTEGER(0..2^16-1)
```

Value assignment: Unsigned binary. Value in km in the operational range 0 to 9 999 km.

2.61. **DriverCardApplicationIdentification**

Information, stored in a driver card related to the identification of the application of the card (Annex 1C requirement 253 and 278).

Generation 1:

typeOfTachographCardId is specifying the implemented type of card.

cardStructureVersion is specifying the the version of the structure that is implemented in the card.

noOfEventsPerType is the number of events per type of event the card can record.

noOfFaultsPerType is the number of faults per type of fault the card can record.

activityStructureLength indicates the number of bytes available for storing activity records.

noOfCardVehicleRecords is the number of vehicle records the card can contain.

noOfCardPlaceRecords is the number of places the card can record.

Generation 2:

[[F2]In addition to generation 1 the following data elements are used:

noOfGNSSADRecords is the number of GNSS accumulated driving records the card can store.

noOfSpecificConditionRecords is the number of specific condition records the card can store.

noOfCardVehicleUnitRecords is the number of vehicle units used records the card can store.**]**

2.62. **DriverCardHolderIdentification**

Information, stored in a driver card, related to the identification of the cardholder (Annex 1C requirement 256 and 281).

```
DriverCardHolderIdentification ::= SEQUENCE {
   cardHolderName                    HolderName,
   cardHolderBirthDate               Datef,
   cardHolderPreferredLanguage       Language
}
```

cardHolderName is the name and first name(s) of the holder of the Driver Card.

cardHolderBirthDate is the date of birth of the holder of the Driver Card.

cardHolderPreferredLanguage is the preferred language of the card holder.

[[F2]2.63. **Reserved for future use]**

2.64. **EGFCertificate**

Generation 2:

Certificate of the external GNSS facility public key for mutual authentication with a VU. The structure of this certificate is specified in Appendix 11.

```
EGFCertificate ::= Certificate
```

2.65. **EmbedderIcAssemblerId**

Provides information about the IC embedder.

```
EmbedderIcAssemblerId ::= SEQUENCE{
   countryCode                       IA5String(SIZE(2)),
   moduleEmbedder                    BCDString(SIZE(2)),
   manufacturerInformation           OCTET STRING(SIZE(1))
}
```

countryCode is the 2 letter country code of the module embedder according to ISO 3166.

moduleEmbedder identifies the module embedder.

manufacturerInformation for manufacturer internal usage.

2.66. **EntryTypeDailyWorkPeriod**

Code to distinguish between begin and end for an entry of a daily work period place and condition of the entry.

Generation 1

Value assignment: according to ISO/IEC8824-1.

Generation 2

Value assignment: according to ISO/IEC8824-1.

2.67. **EquipmentType**

Code to distinguish different types of equipment for the tachograph application.

```
EquipmentType ::= INTEGER(0..255)
```

Generation 1:

Value assignment: According to ISO/IEC8824-1.

Value 0 is reserved for the purpose of designating a Member State or Europe in the CHA field of certificates.

Generation 2:

[[F2]The same values as in generation 1 are used with the following additions:

Note 1: The generation 2 values for the Plaque, Adapter and the External GNSS connection as well as the generation 1 values for the Vehicle Unit and Motion Sensor may be used in SealRecord, i.e. if applicable.

Note 2: In the CardHolderAuthorisation (CHA) field of a generation 2 certificate, the values (1), (2), and (6) are to be interpreted as indicating a certificate for Mutual Authentication for the respective equipment type. For indicating the respective certificate for creating a digital signature, the values (17), (18) or (19) must be used.**]**

2.68. **EuropeanPublicKey**

Generation 1:

The European public key.

```
EuropeanPublicKey ::= PublicKey
```

2.69. **EventFaultRecordPurpose**

Code explaining why an event or a fault has been recorded.

```
EventFaultRecordPurpose ::= OCTET STRING (SIZE(1))
```

Value assignment:

'00'H	one of the 10 most recent (or last) events or faults
'01'H	the longest event for one of the last 10 days of occurrence
'02'H	one of the 5 longest events over the last 365 days
'03'H	the last event for one of the last 10 days of occurrence
'04'H	the most serious event for one of the last 10 days of occurrence
'05'H	one of the 5 most serious events over the last 365 days
'06'H	the first event or fault having occurred after the last calibration
'07'H	an active/on-going event or fault
'08'H to '7F'H	RFU
'80'H to 'FF'H	manufacturer specific

2.70. **EventFaultType**

Code qualifying an event or a fault.

```
EventFaultType ::= OCTET STRING (SIZE(1))
```

Value assignment:

Generation 1:

'0x'H '00'H '01'H '02'H '03'H '04'H '05'H '06'H '07'H '08'H '09'H '0A'H '0B' to '0F'H	General events, No further details, Insertion of a non valid card, Card conflict, Time overlap, Driving without an appropriate card, Card insertion while driving, Last card session not correctly closed, Over speeding, Power supply interruption, Motion data error, Vehicle Motion Conflict, RFU,
'1x'H '10'H '11'H '12'H '13'H '14'H '15'H '16'H '17'H '18'H '19'H to '1F'H	Vehicle unit related security breach attempt events, No further details, Motion sensor authentication failure, Tachograph card authentication failure, Unauthorised change of motion sensor, Card data input integrity error Stored user data integrity error, Internal data transfer error, Unauthorised case opening, Hardware sabotage, RFU,
'2x'H '20'H '21'H '22'H '23'H '24'H '25'H '26'H to '2F'H	Sensor related security breach attempt events, No further details, Authentication failure, Stored data integrity error, Internal data transfer error, Unauthorised case opening, Hardware sabotage, RFU,
'3x'H '30'H '31'H '32'H '33'H '34'H '35'H '36'H to '3F'H	Recording equipment faults, No further details, VU internal fault, Printer fault, Display fault, Downloading fault, Sensor fault, RFU,

'4x'H '40'H '41'H to '4F'H	Card faults, No further details, RFU,
'50'H to '7F'H	RFU,
'80'H to 'FF'H	Manufacturer specific.

[F2Generation 2:

'0x'H '00'H '01'H '02'H '03'H '04'H '05'H '06'H '07'H '08'H '09'H '0A'H '0B'H '0C'H '0D'H '0E'H '0F'H	General events, No further details, Insertion of a non valid card, Card conflict, Time overlap, Driving without an appropriate card, Card insertion while driving, Last card session not correctly closed, Over speeding, Power supply interruption, Motion data error, Vehicle Motion Conflict, Time conflict (GNSS versus VU internal clock), Communication error with the remote communication facility, Absence of position information from GNSS receiver, Communication error with the external GNSS facility, RFU,
'1x'H '10'H '11'H '12'H '13'H '14'H '15'H '16'H '17'H '18'H '19'H '1A'H '1B'H '1C'H to '1F'H	Vehicle unit related security breach attempt events, No further details, Motion sensor authentication failure, Tachograph card authentication failure, Unauthorised change of motion sensor, Card data input integrity error Stored user data integrity error, Internal data transfer error, Unauthorised case opening, Hardware sabotage, Tamper detection of GNSS, External GNSS facility authentication failure, External GNSS facility certificate expired, RFU,

'xx'H '2x'H '20'H '21'H '22'H '23'H '24'H '25'H '26'H to '2F'H	Sensor related security breach attempt events, No further details, Authentication failure, Stored data integrity error, Internal data transfer error, Unauthorised case opening, Hardware sabotage, RFU,
'xx'H '3x'H '30'H '31'H '32'H '33'H '34'H '35'H '36'H '37'H '38'H '39'H '3A'H to '3F'H	Recording equipment faults, No further details, VU internal fault, Printer fault, Display fault, Downloading fault, Sensor fault, Internal GNSS receiver, External GNSS facility, Remote communication facility, ITS interface, RFU,
'xx'H '4x'H '40'H '41'H to '4F'H	Card faults, No further details, RFU,
'50'H to '7F'H	RFU,
'80'H to 'FF'H	Manufacturer specific.]

[F2 2.71. ExtendedSealIdentifier

Generation 2:

The extended seal identifier uniquely identifies a seal (Annex IC requirement 401).

```
ExtendedSealIdentifier ::= SEQUENCE{
   manufacturerCode                OCTET STRING (SIZE(2)),
   sealIdentifier                  OCTET STRING (SIZE(8))
}
```

manufacturerCode is a code of the manufacturer of the seal.

sealIdentifier is an identifier for the seal which is unique for the manufacturer.**]**

2.72. **ExtendedSerialNumber**

Unique identification of an equipment. It can also be used as an equipment Public Key Identifier.

Generation 1:

serialNumber is a serial number for the equipment, unique for the manufacturer, the equipment's type and the month and year below.

monthYear is the identification of the month and the year of manufacturing (or of serial number assignment).

Value assignment: BCD coding of Month (two digits) and Year (two last digits).

type is an identifier of the type of equipment.

Value assignment: manufacturer specific, with 'FFh' reserved value.

manufacturerCode: is the numerical code identifying a manufacturer of type approved equipment.

Generation 2:

serialNumber see Generation 1

monthYear see Generation 1

type indicates the type of equipment

manufacturerCode: see Generation 1.

2.73. **FullCardNumber**

Code fully identifying a tachograph card.

```
FullCardNumber ::= SEQUENCE {
   cardType                              EquipmentType,
   cardIssuingMemberState                NationNumeric,
   cardNumber                            CardNumber
}
```

cardType is the type of the tachograph card.

cardIssuingMemberState is the code of the Member State having issued the card.

cardNumber is the card number.

2.74. **FullCardNumberAndGeneration**

Generation 2:

Code fully identifying a tachograph card and its generation.

```
FullCardNumberAndGeneration ::= SEQUENCE {
   fullCardNumber                        FullCardNumber,
   generation                            Generation
}
```

fullcardNumber identifies the tachograph card.

generation indicates the generation of the tachograph card used.

2.75. **Generation**

Generation 2:

Indicates the generation of tachograph used.

```
Generation ::= INTEGER(0..255)
```

Value assignment:

'00'H	RFU
'01'H	Generation 1
'02'H	Generation 2
'03'H .. 'FF'H	RFU

2.76. **GeoCoordinates**

Generation 2:

The geo-coordinates are encoded as integers. These integers are multiples of the ±DDMM.M encoding for the latitude and ±DDDMM.M for the longitude. Here ±DD respectively ±DDD denotes the degrees and MM.M the minutes.

```
GeoCoordinates ::= SEQUENCE {
   latitude                                INTEGER(-90000..90001),
   longitude                               INTEGER(-180000..180001)
}
```

latitude is encoded as a multiple (factor 10) of the ±DDMM.M representation.

longitude is encoded as a multiple (factor 10) of the ±DDDMM.M representation.

2.77. **GNSSAccuracy**

Generation 2:

The accuracy of the GNSS position data (definition eee)). This accuracy is encoded as integer and is a multiple (factor 10) of the X.Y value provided by the GSA NMEA sentence.

```
GNSSAccuracy ::= INTEGER(1..100)
```

[F2 2.78. **GNSSAccumulatedDriving**

Generation 2:

Information, stored in a driver or workshop card, related to the GNSS position of the vehicle if the accumulated driving time reaches a multiple of three hours (Annex IC requirement 306 and 354).

```
GNSSAccumulatedDriving := SEQUENCE {
   gnssADPointerNewestRecord       INTEGER(0..NoOfGNSSADRecords -1),
   gnssAccumulatedDrivingRecords   SET SIZE(NoOfGNSSADRecords) OF
                                   GNSSAccumulatedDrivingRecord
}
```

gnssADPointerNewestRecord is the index of the last updated GNSS accumulated driving record.

Value assignment is the number corresponding to the numerator of the GNSS accumulated driving record, beginning with '0' for the first occurrence of the GNSS accumulated driving record in the structure.

gnssAccumulatedDrivingRecords is the set of records containing the date and time the accumulated driving reaches a multiple of three hours and information on the position of the vehicle.

2.79. **GNSSAccumulatedDrivingRecord**

Generation 2:

Information, stored in a driver or workshop card, related to the GNSS position of the vehicle if the accumulated driving time reaches a multiple of three hours (Annex IC requirement 305 and 353)

```
GNSSAccumulatedDrivingRecord ::= SEQUENCE {
   timeStamp                    TimeReal,
   gnssPlaceRecord              GNSSPlaceRecord,
   vehicleOdometerValue         OdometerShort
}
```

timeStamp is the date and time when the accumulated driving time reaches a multiple of three hours.

gnssPlaceRecord contains information related to the position of the vehicle.

vehicleOdometerValue is the odometer value when the accumulated driving time reaches a multiple of three hours.**]**

2.80. **GNSSPlaceRecord**

Generation 2:

Information related to the GNSS position of the vehicle (Annex 1C requirements 108, 109, 110, 296, 305, 347, and 353).

```
GNSSPlaceRecord ::= SEQUENCE {
   timeStamp                              TimeReal,
   gnssAccuracy                           GNSSAccuracy,
   geoCoordinates                         GeoCoordinates
}
```

timeStamp is the date and time when the GNSS position of the vehicle was determined.

gnssAccuracy is the accuracy of the GNSS position data.

geoCoordinates is the recorded location using GNSS.

2.81. **HighResOdometer**

Odometer value of the vehicle: Accumulated distance travelled by the vehicle during its operation.

```
HighResOdometer ::= INTEGER(0..2^32-1)
```

Value assignment: Unsigned binary. Value in 1/200 km in the operating range 0 to 21 055 406 km.

2.82. **HighResTripDistance**

A distance travelled during all or part of a journey.

```
HighResTripDistance ::= INTEGER(0..2^32-1)
```

Value assignment: Unsigned binary. Value in 1/200 km in the operating range 0 to 21 055 406 km.

2.83. **HolderName**

The surname and first name(s) of a card holder.

```
HolderName ::= SEQUENCE {
   holderSurname                    Name,
   holderFirstNames                 Name
}
```

holderSurname is the surname (family name) of the holder. This surname does not include titles.

Value assignment: When a card is not personal, holderSurname contains the same information as companyName or workshopName or controlBodyName.

holderFirstNames is the first name(s) and initials of the holder.

2.84. **InternalGNSSReceiver**

Generation 2:

Information if the GNSS receiver is internal or external to the vehicle unit. True means that the GNSS receiver is internal to the VU. False means that the GNSS receiver is external.

```
InternalGNSSReceiver ::= BOOLEAN
```

2.85. **K-ConstantOfRecordingEquipment**

Constant of the recording equipment (definition m)).

```
K-ConstantOfRecordingEquipment ::= INTEGER(0..2^16-1)
```

Value assignment: Pulses per kilometer in the operating range 0 to 64 255 pulses/km.

[F2 2.86. **KeyIdentifier**

A unique identifier of a Public Key used to reference and select the key. It also identifies the holder of the key.

```
KeyIdentifier ::= CHOICE {
   extendedSerialNumber          ExtendedSerialNumber,
   certificateRequestID          CertificateRequestID,
   certificationAuthorityKID     CertificationAuthorityKID
}
```

The first choice is suitable to reference the public key of a Vehicle Unit, of a tachograph card or of an external GNSS facility.

The second choice is suitable to reference the public key of a Vehicle Unit (in cases where the serial number of the Vehicle Unit cannot be known at certificate generation time).

The third choice is suitable to reference the public key of a Member State.**]**

2.87. **KMWCKey**

Generation 2:

AES key and its associated key version used for VU — Motion Sensor pairing. For details see Appendix 11.

```
KMWCKey ::= SEQUENCE {
   kMWCKey                                 AESKey,
   keyVersion                              INTEGER (SIZE(1))
}
```

kMWCKey is the length of the AES key concatenated with the key which is used for VU — Motion Sensor pairing.

keyVersion denotes the key version of the AES key.

2.88. **Language**

Code identifying a language.

```
Language ::= IA5String(SIZE(2))
```

Value assignment: Two-letter lower-case coding according to ISO 639.

2.89. **LastCardDownload**

Date and time, stored on a driver card, of last card download (for other purposes than control) Annex 1C requirement 257 and 282. This date is updateable by a VU or any card reader.

```
LastCardDownload ::= TimeReal
```

Value assignment: not further specified.

2.90. **LinkCertificate**

Generation 2:

The link certificate between European Root CA key pairs.

```
LinkCertificate ::= Certificate
```

2.91. **L-TyreCircumference**

Effective circumference of the wheel tyres (definition u)).

```
L-TyreCircumference ::= INTEGER(0.. 2^16-1)
```

Value assignment: Unsigned binary, value in 1/8 mm in the operating range 0 to 8 031 mm.

[[F2]2.92. **MAC**

Generation 2:

A cryptographic check sum of 8, 12 or 16 bytes length corresponding to the cipher suites specified in Appendix 11.**]**

```
MAC ::= CHOICE {
   Mac8                    OCTET STRING (SIZE(8)),
   Mac12                   OCTET STRING (SIZE(12)),
   Mac16                   OCTET STRING (SIZE(16)),
}';
```

2.93. **ManualInputFlag**

Code identifying whether a cardholder has manually entered driver activities at card insertion or not (Annex 1B requirement 081 and Annex 1C requirement 102).

```
ManualInputFlag ::= INTEGER {
   noEntry                                 (0)
   manualEntries                           (1)
}
```

Value assignment: not further specified.

2.94. **ManufacturerCode**

Code identifying a manufacturer of type approved equipment.

```
ManufacturerCode ::= INTEGER(0..255)
```

The laboratory competent for interoperability tests maintains and publishes the list of manufacturer codes on its web site (Annex 1C requirement 454).

ManufacturerCodes are provisionally assigned to developers of tachograph equipment on application to the laboratory competent for interoperability tests.

2.95. **ManufacturerSpecificEventFaultData**

Generation 2:

Manufacturer specific error codes simplify the error analysis and maintenance of vehicle units.

```
ManufacturerSpecificEventFaultData ::= SEQUENCE {
   manufacturerCode                    ManufacturerCode,
   manufacturerSpecificErrorCode       OCTET STRING(SIZE(3))
}
```

manufacturerCode identifies the manufacturer of the Vehicle Unit.

manufacturerSpecificErrorCode is an error code specific to the manufacturer.

2.96. **MemberStateCertificate**

The certificate of the public key of a member state issued by the European certification authority.

```
MemberStateCertificate ::= Certificate
```

2.97. **MemberStateCertificateRecordArray**

Generation 2:

The member state certificate plus metadata as used in the download protocol.

```
MemberStateCertificateRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       MemberStateCertificate
}
```

recordType denotes the type of the record (MemberStateCertificate). **Value Assignment**: See RecordType

recordSize is the size of the MemberStateCertificate in bytes.

noOfRecords is the number of records in the set records. The value shall be set to 1 as the certficates may have different lengths.

records is the set of member state certificates.

2.98. **MemberStatePublicKey**

Generation 1:

The public key of a Member State.

```
MemberStatePublicKey ::= PublicKey
```

2.99. **Name**

A name.

```
Name ::= SEQUENCE {
   codePage                            INTEGER (0..255),
   name                                OCTET STRING (SIZE(35))
}
```

codePage specifies a character set defined in Chapter 4,

name is a name encoded using the specified character set.

2.100. **NationAlpha**

Alphabetic reference to a country shall be in accordance with the distinguishing signs used on vehicles in international traffic (United Nations Vienna Convention on Road Traffic, 1968).

```
NationAlpha ::= IA5String(SIZE(3))
```

The Nation Alpha and Numeric codes shall be held on a list maintained on the website of the laboratory appointed to carry out interoperability testing, as set out in Annex 1C requirement 440.

2.101. **NationNumeric**

Numerical reference to a country.

```
NationNumeric ::= INTEGER(0 .. 255)
```

Value assignment: see data type 2.100 (NationAlpha).

Any amendment or updating of the Nation Alpha or Numeric specification described in the above paragraph shall only be made out after the appointed laboratory has obtained the views of type approved digital and smart tachograph vehicle unit manufacturers.

2.102. **NoOfCalibrationRecords**

Number of calibration records, a workshop card can store.

Generation 1:

Value assignment: see Appendix 2.

Generation 2:

Value assignment: see Appendix 2.

2.103. **NoOfCalibrationsSinceDownload**

Counter indicating the number of calibrations performed with a workshop card since its last download (Annex 1C requirement 317 and 340).

```
NoOfCalibrationsSinceDownload ::= INTEGER(0..2^16-1)
```

Value assignment: Not specified further.

2.104. **NoOfCardPlaceRecords**

Number of place records a driver or workshop card can store.

Generation 1:

Value assignment: see Appendix 2.

Generation 2:

Value assignment: see Appendix 2.

2.105. **NoOfCardVehicleRecords**

Number of vehicles used records a driver or workshop card can store.

```
NoOfCardVehicleRecords ::= INTEGER(0.. 2^16-1)
```

Value assignment: see Appendix 2.

2.106. **NoOfCardVehicleUnitRecords**

Generation 2:

Number of vehicle units used records a driver or workshop card can store.

```
NoOfCardVehicleUnitRecords ::= INTEGER(0.. 2^16-1)
```

Value assignment: see Appendix 2.

2.107. **NoOfCompanyActivityRecords**

Number of company activity records, a company card can store.

```
NoOfCompanyActivityRecords ::= INTEGER(0.. 2^16-1)
```

Value assignment: see Appendix 2.

2.108. **NoOfControlActivityRecords**

Number of control activity records, a control card can store.

```
NoOfControlActivityRecords ::= INTEGER(0.. 2^16-1)
```

Value assignment: see Appendix 2.

2.109. **NoOfEventsPerType**

Number of events per type of event a card can store.

```
NoOfEventsPerType ::= INTEGER(0..255)
```

Value assignment: see Appendix 2.

2.110. **NoOfFaultsPerType**

Number of faults per type of fault a card can store.

```
NoOfFaultsPerType ::= INTEGER(0..255)
```

Value assignment: see Appendix 2.

[[F2]2.111. **NoOfGNSSADRecords**

Generation 2:

Number of GNSS accumulated driving records a card can store.

$$\text{NoOfGNSSADRecords ::= INTEGER (0..2}^{16}\text{-1)}$$

Value assignment: see Appendix 2.**]**

2.112. **NoOfSpecificConditionRecords**

Generation 2:

Number of specific condition records a card can store.

$$\text{NoOfSpecificConditionRecords ::= INTEGER(0..2}^{16}\text{-1)}$$

Value assignment: see Appendix 2.

2.113. **OdometerShort**

Odometer value of the vehicle in a short form.

$$\text{OdometerShort ::= INTEGER(0..2}^{24}\text{-1)}$$

Value assignment: Unsigned binary. Value in km in the operating range 0 to 9 999 999 km.

2.114. **OdometerValueMidnight**

The vehicle's odometer value at midnight on a given day (Annex 1B requirement 090 and Annex 1C requirement 113).

```
OdometerValueMidnight ::= OdometerShort
```

Value assignment: not further specified.

2.115. **OdometerValueMidnightRecordArray**

Generation 2:

The OdometerValueMidnight plus metadata used in the download protocol.

```
OdometerValueMidnightRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       OdometerValueMidnight
}
```

recordType denotes the type of the record (OdometerValueMidnight). **Value Assignment**: See RecordType

recordSize is the size of the OdometerValueMidnight in bytes.

noOfRecords is the number of records in the set records.

records is the set of OdometerValueMidnight records.

2.116. **OverspeedNumber**

Number of over speeding events since the last over speeding control.

```
OverspeedNumber ::= INTEGER(0..255)
```

Value assignment: 0 means that no over speeding event has occurred since the last over speeding control, 1 means that one over speeding event has occurred since the last over speeding control …255 means that 255 or more over speeding events have occurred since the last over speeding control.

2.117. **PlaceRecord**

Information related to a place where a daily work period begins or ends (Annex 1C requirements 108, 271, 296, 324, and 347).

Generation 1:

entryTime is a date and time related to the entry.

entryTypeDailyWorkPeriod is the type of entry.

dailyWorkPeriodCountry is the country entered.

dailyWorkPeriodRegion is the region entered.

vehicleOdometerValue is the odometer value at the time of place entry.

Generation 2:

In addition to Generation 1 the following component is used:

entryGNSSPlaceRecord is the recorded location and time.

2.118. **PreviousVehicleInfo**

Information related to the vehicle previously used by a driver when inserting his card in a vehicle unit (Annex 1B requirement 081 and Annex 1C requirement 102).

Generation 1:

vehicleRegistrationIdentification is the VRN and the registering Member State of the vehicle.

cardWithdrawalTime is the card withdrawal date and time.
Generation 2:

In addition to generation 1 the following data element is used:

vuGeneration identifies the generation of the vehicle unit.

2.119. **PublicKey**

Generation 1:

A public RSA key.

```
PublicKey ::= SEQUENCE {
   rsaKeyModulus                    RSAKeyModulus,
   rsaKeyPublicExponent             RSAKeyPublicExponent
}
```

rsaKeyModulus is the Modulus of the key pair.

rsaKeyPublicExponent is the public exponent of the key pair.

2.120. **RecordType**

Generation 2:

Reference to a record type. This data type is used in RecordArrays.

```
RecordType ::= OCTET STRING(SIZE(1))
```

Value assignment:

'01'H	ActivityChangeInfo,
'02'H	CardSlotsStatus,
'03'H	CurrentDateTime,
'04'H	MemberStateCertificate,
'05'H	OdometerValueMidnight,
'06'H	DateOfDayDownloaded,
'07'H	SensorPaired,
'08'H	Signature,
'09'H	SpecificConditionRecord,
'0A'H	VehicleIdentificationNumber,
'0B'H	VehicleRegistrationNumber,
'0C'H	VuCalibrationRecord,
'0D'H	VuCardIWRecord,
'0E'H	VuCardRecord,
'0F'H	VuCertificate,
'10'H	VuCompanyLocksRecord,
'11'H	VuControlActivityRecord,
'12'H	VuDetailedSpeedBlock,
'13'H	VuDownloadablePeriod,
'14'H	VuDownloadActivityData,
'15'H	VuEventRecord,
▶(1)'16'H◀	[F2VuGNSSADRecord,]
'17'H	VuITSConsentRecord,
'18'H	VuFaultRecord,
'19'H	VuIdentification,
'1A'H	VuOverSpeedingControlData,
'1B'H	VuOverSpeedingEventRecord,
'1C'H	VuPlaceDailyWorkPeriodRecord,
'1D'H	VuTimeAdjustmentGNSSRecord,
'1E'H	VuTimeAdjustmentRecord,
'1F'H	VuPowerSupplyInterruptionRecord,
'20'H	SensorPairedRecord,
'21'H	SensorExternalGNSSCoupledRecord,
'22'H to '7F'H	RFU,
'80'H to 'FF'H	Manufacturer specific.

2.121. **RegionAlpha**

Alphabetic reference to a region within a specified country.

```
RegionAlpha ::= IA5STRING(SIZE(3))
```

Generation 1:

Value assignment:

Generation 2:

The RegionAlpha codes shall be held on a list maintained on the website of the laboratory appointed to carry out interoperability testing.

2.122. **RegionNumeric**

Numerical reference to a region within a specified country.

```
RegionNumeric ::= OCTET STRING (SIZE(1))
```

Generation 1:

Value assignment:

Generation 2:

The RegionNumeric codes shall be held on a list maintained on the website of the laboratory appointed to carry out interoperability testing.

2.123. **RemoteCommunicationModuleSerialNumber**

Generation 2:

Serial number of the Remote Communication Module.

```
RemoteCommunicationModuleSerialNumber ::= ExtendedSerialNumber
```

2.124. **RSAKeyModulus**

Generation 1:

The modulus of a RSA key pair.

```
RSAKeyModulus ::= OCTET STRING (SIZE(128))
```

Value assignment: Unspecified.

2.125. **RSAKeyPrivateExponent**

Generation 1:

The private exponent of a RSA key pair.

```
RSAKeyPrivateExponent ::= OCTET STRING (SIZE(128))
```

Value assignment: Unspecified.

2.126. **RSAKeyPublicExponent**

Generation1:

The public exponent of a RSA key pair.

```
RSAKeyPublicExponent ::= OCTET STRING (SIZE(8))
```

Value assignment: Unspecified.

2.127. **RtmData**

Generation2:

For the definition of this data type see Appendix 14.

2.128. **SealDataCard**

Generation 2:

This data type stores information about the seals that are attached to the different components of a vehicle and is intended for storage on a card. This data type is related to Annex 1C requirement 337.

```
SealDataCard ::= SEQUENCE {
   noOfSealRecords                    INTEGER(1..5),
   sealRecords                        SET SIZE(noOfSealRecords) OF SealRecord
}
```

noOfSealRecords is the number of records in sealRecords.

sealRecords is a set of seal records.

2.129. **SealDataVu**

Generation 2:

This data type stores information about the seals that are attached to the different components of a vehicle and is intended for storage in a Vehicle Unit.

```
SealDataVu ::= SEQUENCE SIZE(5) OF {
   sealRecords                              SealRecord
}
```

sealRecords is a set of seal records. If there are less than 5 seals available the value of the EquipmentType in all unused sealRecords shall be set to 16, i.e. unused.

2.130. **SealRecord**

Generation 2:

This data type stores information about a seal that is attached to a component. This data type is related to Annex 1C requirement 337.

```
SealRecord ::= SEQUENCE {
   equipmentType                        EquipmentType,
   extendedSealIdentifier               ExtendedSealIdentifier
}
```

equipmentType identifies the type of equipment the seal is attached to.

extendedSealIdentifier is the identifier of the seal attached to the equipment.

2.131. **SensorApprovalNumber**

Type approval number of the sensor.

Generation 1:

Value assignment: Unspecified.

Generation 2:

Value assignment:

The approval number shall be provided as published on the corresponding European Commission web site, i.e. for example including hyphens if any. The approval number shall be left-aligned.

2.132. **SensorExternalGNSSApprovalNumber**

Generation 2:

Type approval number of the external GNSS facility.

```
SensorExternalGNSSApprovalNumber ::= IA5String(SIZE(16))
```

Value assignment:

The approval number shall be provided as published on the corresponding European Commission web site, i.e. for example including hyphens if any. The approval number shall be left-aligned.

2.133. **SensorExternalGNSSCoupledRecord**

Generation 2:

Information, stored in a vehicle unit, related to the identification of the external GNSS facility coupled with the vehicle unit (Annex 1C requirement 100).

```
SensorExternalGNSSCoupledRecord ::= SEQUENCE {
   sensorSerialNumber                  SensorGNSSSerialNumber,
   sensorApprovalNumber                SensorExternalGNSSApprovalNumber,
   sensorCouplingDate                  SensorGNSSCouplingDate
}
```

sensorSerialNumber is the serial number of the external GNSS facility coupled with the vehicle unit.

sensorApprovalNumber is the approval number of this external GNSS facility.

sensorCouplingDate is a date of coupling of this external GNSS facility with the vehicle unit.

2.134. **SensorExternalGNSSIdentification**

Generation 2:

Information related to the identification of the external GNSS facility (Annex 1C requirement 98).

```
SensorExternalGNSSIdentification ::= SEQUENCE {

   sensorSerialNumber                  SensorGNSSSerialNumber,

   sensorApprovalNumber                SensorExternalGNSSApprovalNumber,

   sensorSCIdentifier                  SensorExternalGNSSSCIdentifier,

   sensorOSIdentifier                  SensorExternalGNSSOSIdentifier

}
```

sensorSerialNumber is the extended serial number of the external GNSS facility.

sensorApprovalNumber is the approval number of the external GNSS facility.

sensorSCIdentifier is the identifier of the security component of the external GNSS facility.

sensorOSIdentifier is the identifier of the operating system of the external GNSS facility.

2.135. **SensorExternalGNSSInstallation**

Generation 2:

Information, stored in an external GNSS facility, related to the installation of the external GNSS sensor (Annex 1C requirement 123).

```
SensorExternalGNSSInstallation ::= SEQUENCE {
   sensorCouplingDateFirst              SensorGNSSCouplingDate,
   firstVuApprovalNumber                VuApprovalNumber,
   firstVuSerialNumber                  VuSerialNumber,
   sensorCouplingDateCurrent            SensorGNSSCouplingDate,
   currentVuApprovalNumber              VuApprovalNumber,
   currentVUSerialNumber                VuSerialNumber
}
```

sensorCouplingDateFirst is the date of the first coupling of external GNSS facility with a vehicle unit.

firstVuApprovalNumber is the approval number of the first vehicle unit coupled with the external GNSS facility.

firstVuSerialNumber is the serial number of the first vehicle unit paired with the external GNSS facility.

sensorCouplingDateCurrent is the date of the current coupling of external GNSS facility with a vehicle unit.

currentVuApprovalNumber is the approval number of the vehicle unit currently coupled with the external GNSS facility.

currentVUSerialNumber is the serial number of the vehicle unit currently coupled with the external GNSS facility.

2.136. **SensorExternalGNSSOSIdentifier**

Generation 2:

Identifier of the operating system of the external GNSS facility.

```
SensorOSIdentifier ::= IA5String(SIZE(2))
```

Value assignment: manufacturer specific.

2.137. **SensorExternalGNSSSCIdentifier**

Generation 2:

This type is used e.g. to identify the cryptographic module of the external GNSS facility.

Identifier of the security component of the external GNSS facility.

```
SensorExternalGNSSSCIdentifier ::= IA5String(SIZE(8))
```

Value assignment: component manufacturer specific.

2.138. **SensorGNSSCouplingDate**

Generation 2:

Date of a coupling of the external GNSS facility with a vehicle unit.

```
SensorGNSSCouplingDate ::= TimeReal
```

Value assignment: Unspecified.

2.139. **SensorGNSSSerialNumber**

Generation 2:

This type is used to store the serial number of the GNSS receiver both when it is inside the VU and when it is outside the VU.

Serial number of the GNSS receiver.

```
SensorGNSSSerialNumber ::= ExtendedSerialNumber
```

2.140. **SensorIdentification**

Information, stored in a motion sensor, related to the identification of the motion sensor (Annex 1B requirement 077 and Annex 1C requirement 95).

```
SensorIdentification ::= SEQUENCE {
   sensorSerialNumber                SensorSerialNumber,
   sensorApprovalNumber              SensorApprovalNumber,
   sensorSCIdentifier                SensorSCIdentifier,
   sensorOSIdentifier                SensorOSIdentifier
}
```

sensorSerialNumber is the extended serial number of the motion sensor (includes part number and manufacturer code).

sensorApprovalNumber is the approval number of the motion sensor.

sensorSCIdentifier is the identifier of the security component of the motion sensor.

sensorOSIdentifier is the identifier of the operating system of the motion sensor.

2.141. **SensorInstallation**

Information, stored in a motion sensor, related to the installation of the motion sensor (Annex 1B requirement 099 and Annex 1C requirement 122).

```
SensorInstallation ::= SEQUENCE {
   sensorPairingDateFirst            SensorPairingDate,
   firstVuApprovalNumber             VuApprovalNumber,
   firstVuSerialNumber               VuSerialNumber,
   sensorPairingDateCurrent          SensorPairingDate,
   currentVuApprovalNumber           VuApprovalNumber,
   currentVUSerialNumber             VuSerialNumber
}
```

sensorPairingDateFirst is the date of the first pairing of the motion sensor with a vehicle unit.

firstVuApprovalNumber is the approval number of the first vehicle unit paired with the motion sensor.

firstVuSerialNumber is the serial number of the first vehicle unit paired with the motion sensor.

sensorPairingDateCurrent is the date of the current pairing of the motion sensor with the vehicle unit.

currentVuApprovalNumber is the approval number of the vehicle unit currently paired with the motion sensor.

currentVUSerialNumber is the serial number of the vehicle unit currently paired with the motion sensor.

2.142. **SensorInstallationSecData**

Information, stored in a workshop card, related to the security data needed for pairing motion sensors to vehicle units (Annex 1C requirement 308 and 331).

Generation 1:

Value assignment: in accordance with ISO 16844-3.

Generation 2:

As described in Appendix 11 a workshop card shall store up to three keys for VU Motion Sensor pairing. These keys have different key versions.

2.143. **SensorOSIdentifier**

Identifier of the operating system of the motion sensor.

```
SensorOSIdentifier ::= IA5String(SIZE(2))
```

Value assignment: manufacturer specific.

2.144. **SensorPaired**

Generation 1:

Information, stored in a vehicle unit, related to the identification of the motion sensor paired with the vehicle unit (Annex 1B requirement 079).

```
SensorPaired ::= SEQUENCE {
   sensorSerialNumber                SensorSerialNumber,
   sensorApprovalNumber              SensorApprovalNumber,
   sensorPairingDateFirst            SensorPairingDate
}
```

sensorSerialNumber is the serial number of the motion sensor currently paired with the vehicle unit.

sensorApprovalNumber is the approval number of the motion sensor currently paired with the vehicle unit.

sensorPairingDateFirst is the date of the first pairing with a vehicle unit of the motion sensor currently paired with the vehicle unit.

2.145. **SensorPairedRecord**

Generation 2:

Information, stored in a vehicle unit, related to the identification of a motion sensor paired with the vehicle unit (Annex 1C requirement 97).

```
SensorPairedRecord ::= SEQUENCE {
   sensorSerialNumber                    SensorSerialNumber,
   sensorApprovalNumber                  SensorApprovalNumber,
   sensorPairingDate                     SensorPairingDate
}
```

sensorSerialNumber is the serial number of a motion sensor paired with the vehicle unit.

sensorApprovalNumber is the approval number of this motion sensor.

sensorPairingDate is a date of pairing of this motion sensor with the vehicle unit.

2.146. **SensorPairingDate**

Date of a pairing of the motion sensor with a vehicle unit.

```
SensorPairingDate ::= TimeReal
```

Value assignment: Unspecified.

2.147. **SensorSCIdentifier**

Identifier of the security component of the motion sensor.

```
SensorSCIdentifier ::= IA5String(SIZE(8))
```

Value assignment: component manufacturer specific.

2.148. **SensorSerialNumber**

Serial number of the motion sensor.

```
SensorSerialNumber ::= ExtendedSerialNumber
```

2.149. **Signature**

A digital signature.

Generation 1:

Value assignment: in accordance with Appendix 11 Common security mechanisms.

Generation 2:

Value assignment: in accordance with Appendix 11 Common security mechanisms.

2.150. **SignatureRecordArray**

Generation 2:

A set of signatures plus metadata used in the download protocol.

```
SignatureRecordArray ::= SEQUENCE {
   recordType                            RecordType,
   recordSize                            INTEGER(1..65535),
   noOfRecords                           INTEGER(0..65535),
   records                               SET SIZE(noOfRecords) OF Signature
}
```

recordType denotes the type of the record (Signature). **Value Assignment**: See RecordType

recordSize is the size of the Signature in bytes.

noOfRecords is the number of records in the set records. The value shall be set to 1 as the signatures may have different lengths.

records is the set of signatures.

2.151. **SimilarEventsNumber**

The number of similar events for one given day (Annex 1B requirement 094 and Annex 1C requirement 117).

```
SimilarEventsNumber ::= INTEGER(0..255)
```

Value assignment: 0 is not used, 1 means that only one event of that type has occurred and has been stored on that day, 2 means that 2 events of that type has occurred on that day (one only has been stored), …255 means that 255 or more events of that type have occurred on that day.

2.152. **SpecificConditionRecord**

Information, stored in a driver card, a workshop card or a vehicle unit, related to a specific condition (requirements Annex 1C 130, 276, 301, 328, and 355).

```
SpecificConditionRecord ::= SEQUENCE {
   entryTime                              TimeReal,
   specificConditionType                  SpecificConditionType
}
```

entryTime is the date and time of the entry.

specificConditionType is the code identifying the specific condition.

2.153. **SpecificConditions**

Information, stored in a driver card, a workshop card or a vehicle unit, related to a specific condition (Annex 1C requirement 131, 277, 302, 329, and 356).

Generation 2:

```
SpecificConditions := SEQUENCE {
   conditionPointerNewestRecord         INTEGER(0..NoOfSpecificConditionRecords-1),
   specificConditionRecords             SET SIZE(NoOfSpecificConditionRecords) OF
                                        SpecificConditionRecord
}
```

conditionPointerNewestRecord is the index of the last updated specific condition record.

Value assignment: Number corresponding to the numerator of the specific condition record, beginning with ‘0’ for the first occurrence of the specific condition record in the structure.

specificConditionRecords is the set of records containing information on the specific conditions recorded.

2.154. **SpecificConditionType**

Code identifying a specific condition (Annex 1B requirements 050b, 105a, 212a and 230a and Annex 1C requirements 62).

```
SpecificConditionType ::= INTEGER(0..255)
```

Generation 1:

Value assignment:

‘00’H	RFU
‘01’H	Out of scope — Begin

‘02’H	Out of scope — End
‘03’H	Ferry / Train crossing
‘04’H.. ‘FF’H	RFU

Generation 2:

Value assignment:

‘00’H	RFU
‘01’H	Out of scope — Begin
‘02’H	Out of scope — End
‘03’H	Ferry / Train crossing — Begin
‘04’H	Ferry / Train crossing — End
‘05’H.. ‘FF’H	RFU

2.155. **Speed**

Speed of the vehicle (km/h).

```
Speed ::= INTEGER(0..255)
```

Value assignment: kilometers per hour in the operational range 0 to 220 km/h.

2.156. **SpeedAuthorised**

Maximum authorised Speed of the vehicle (definition hh)).

```
SpeedAuthorised ::= Speed
```

2.157. **SpeedAverage**

Average speed in a previously defined duration (km/h).

```
SpeedAverage ::= Speed
```

2.158. **SpeedMax**

Maximum speed measured in a previously defined duration.

```
SpeedMax ::= Speed
```

2.159. **TachographPayload**

Generation 2:

For the definition of this data type see Appendix 14.

[[F2]2.160. **Reserved for future use]**

2.161. **TDesSessionKey**

Generation 1:

A triple DES session key.

```
TDesSessionKey ::= SEQUENCE {
   tDesKeyA                                OCTET STRING (SIZE(8)),
   tDesKeyB                                OCTET STRING (SIZE(8))
}
```

Value assignment: not further specified.

[[F2]2.162. **TimeReal**

Code for a combined date and time field, where the date and time are expressed as seconds past 00h.00m.00s. on 1 January 1970 UTC.

```
TimeReal {INTEGER:TimeRealRange} ::= INTEGER (0..TimeRealRange)
```

Value assignment – **Octet aligned**: Number of seconds since midnight 1 January 1970 UTC.

The max. possible date/time is in the year 2106.**]**

2.163. **TyreSize**

Designation of tyre dimensions.

```
TyreSize ::= IA5String(SIZE(15))
```

Value assignment: in accordance with Directive 92/23 (EEC) 31/03/92 O.J. L129 p.95.

2.164. **VehicleIdentificationNumber**

Vehicle Identification Number (VIN) referring to the vehicle as a whole, normally chassis serial number or frame number.

```
VehicleIdentificationNumber ::= IA5String(SIZE(17))
```

Value assignment: As defined in ISO 3779.

2.165. **VehicleIdentificationNumberRecordArray**

Generation 2:

The Vehicle Idenification Number plus metadata as used in the download protocol.

```
VehicleIdentificationNumberRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       VehicleIdentificationNumber
}
```

recordType denotes the type of the record (VehicleIdentificationNumber). **Value Assignment**: See RecordType

recordSize is the size of the VehicleIdentificationNumber in bytes.

noOfRecords is the number of records in the set records.

records is the set of vehicle identification numbers.

2.166. **VehicleRegistrationIdentification**

Identification of a vehicle, unique for Europe (VRN and Member State).

```
VehicleRegistrationIdentification ::= SEQUENCE {
   vehicleRegistrationNation            NationNumeric,
   vehicleRegistrationNumber            VehicleRegistrationNumber
}
```

vehicleRegistrationNation is the nation where the vehicle is registered.

vehicleRegistrationNumber is the registration number of the vehicle (VRN).

2.167. **VehicleRegistrationNumber**

Registration number of the vehicle (VRN). The registration number is assigned by the vehicle licensing authority.

```
VehicleRegistrationNumber ::= SEQUENCE {
   codePage                              INTEGER (0..255),
   vehicleRegNumber                      OCTET STRING (SIZE(13))
}
```

codePage specifies a character set defined in Chapter 4,

vehicleRegNumber is a VRN encoded using the specified character set.

Value assignment: Country specific.

2.168. **VehicleRegistrationNumberRecordArray**

Generation 2:

The Vehicle Registration Number plus metadata as used in the download protocol.

```
VehicleRegistrationNumberRecordArray ::= SEQUENCE {
   recordType                            RecordType,
   recordSize                            INTEGER(1..65535),
   noOfRecords                           INTEGER(0..65535),
   records                               SET SIZE(noOfRecords) OF
                                         VehicleRegistrationNumber
}
```

recordType denotes the type of the record (VehicleRegistrationNumber). **Value Assignment**: See RecordType

recordSize is the size of the VehicleRegistrationNumber in bytes.

noOfRecords is the number of records in the set records.

records is the set of vehicle registration numbers.

2.169. **VuAbility**

Generation 2:

Information stored in a VU on the ability of the VU to use generation 1 tachograph cards or not (Annex 1C requirement 121).

```
VuAbility ::= OCTET STRING (SIZE(1))
```

Value assignment — Octet Aligned:‘xxxxxxxa’B (8 bits)

For the ability to support of generation 1:

‘a’B	Ability to support generation 1 tachograph cards: ‘0’ B Generation 1 is supported, ‘1’B Generation1 is not supported,
‘xxxxxxx’B	RFU

2.170. **VuActivityDailyData**

Generation 1:

Information, stored in a VU, related to changes of activity and/or changes of driving status and/or changes of card status for a given calendar day (Annex 1B requirement 084 and Annex 1C requirement 105, 106, 107) and to slots status at 00:00 that day.

```
VuActivityDailyData ::= SEQUENCE {
   noOfActivityChanges                    INTEGER SIZE(0..1440),
   activityChangeInfos                    SET SIZE(noOfActivityChanges) OF
                                          ActivityChangeInfo
}
```

noOfActivityChanges is the number of ActivityChangeInfo words in the activityChangeInfos set.

activityChangeInfos is the set of ActivityChangeInfo words stored in the VU for the day. It always includes two ActivityChangeInfo words giving the status of the two slots at 00:00 that day.

2.171. VuActivityDailyRecordArray

Generation 2:

Information, stored in a VU, related to changes of activity and/or changes of driving status and/or changes of card status for a given calendar day (Annex 1C requirement 105, 106, 107) and to slots status at 00:00 that day.

```
VuActivityDailyRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF ActivityChangeInfo
}
```

recordType denotes the type of the record (ActivityChangeInfo). **Value Assignment**: See RecordType

recordSize is the size of the ActivityChangeInfo in bytes.

noOfRecords is the number of records in the set records.

records is the set of ActivityChangeInfo words stored in the VU for the day. It always includes two ActivityChangeInfo words giving the status of the two slots at 00:00 that day.

2.172. VuApprovalNumber

Type approval number of the vehicle unit.

Generation 1:

Value assignment: Unspecified.

Generation 2:

Value assignment:

The approval number shall be provided as published on the corresponding European Commission web site, i.e. for example including hyphens if any. The approval number shall be left-aligned.

2.173. VuCalibrationData

Generation 1:

Information, stored in a vehicle unit, related to the calibrations of the recording equipment (Annex 1B requirement 098).

```
VuCalibrationData ::= SEQUENCE {
   noOfVuCalibrationRecords              INTEGER(0..255),
   vuCalibrationRecords                  SET SIZE(noOfVuCalibrationRecords) OF
                                         VuCalibrationRecord
}
```

noOfVuCalibrationRecords is the number of records contained in the vuCalibrationRecords set.

vuCalibrationRecords is the set of calibration records.

2.174. **VuCalibrationRecord**

Information, stored in a vehicle unit, related a calibration of the recording equipment (Annex 1B requirement 098 and Annex 1C requirement 119 and 120).

Generation 1:

calibrationPurpose is the purpose of the calibration.

workshopName, workshopAddress are the workshop name and address.

workshopCardNumber identifies the workshop card used during the calibration.

workshopCardExpiryDate is the card expiry date.

vehicleIdentificationNumber is the VIN.

vehicleRegistrationIdentification contains the VRN and registering Member State.

wVehicleCharacteristicConstant is the characteristic coefficient of the vehicle.

kConstantOfRecordingEquipment is the constant of the recording equipment.

lTyreCircumference is the effective circumference of the wheel tyres.

tyreSize is the designation of the dimension of the tyres mounted on the vehicle

authorisedSpeed is the authorised speed of the vehicle.

oldOdometerValue, newOdometerValue are the old and new values of the odometer.

oldTimeValue, newTimeValue are the old and new values of date and time.

nextCalibrationDate is the date of the next calibration of the type specified in CalibrationPurpose to be carried out by the authorised inspection authority.

Generation 2:

In addition to generation 1 the following data element is used:

sealDataVu gives information about the seals that are attached to different components of the vehicle.

2.175. **VuCalibrationRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to the calibrations of the recording equipment (Annex 1C requirement 119 and 120).

```
VuCalibrationRecordArray ::= SEQUENCE {
   recordType                              RecordType,
   recordSize                              INTEGER(1..65535),
   noOfRecords                             INTEGER(0..65535),
   records                                 SET SIZE(noOfRecords) OF
                                           VuCalibrationRecord
}
```

recordType denotes the type of the record (VuCalibrationRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuCalibrationRecord in bytes.

noOfRecords is the number of records in the set records.

records is the set of calibration records.

2.176. **VuCardIWData**

Generation 1:

Information, stored in a vehicle unit, related to insertion and withdrawal cycles of driver cards or of workshop cards in the vehicle unit (Annex 1B requirement 081 and Annex 1C requirement 103).

```
VuCardIWData ::= SEQUENCE {
   noOfIWRecords                    INTEGER(0..2^16-1),
   vuCardIWRecords                  SET SIZE(noOfIWRecords) OF VuCardIWRecord
}
```

noOfIWRecords is the number of records in the set vuCardIWRecords.

vuCardIWRecords is a set of records related to card insertion withdrawal cycles.

2.177. **VuCardIWRecord**

Information, stored in a vehicle unit, related to an insertion and withdrawal cycle of a driver card or of a workshop card in the vehicle unit (Annex 1B requirement 081 and Annex 1C requirement 102).

Generation 1:

cardHolderName is the driver or workshop card holder's surname and first names as stored in the card.

fullCardNumber is the type of card, its issuing Member State and its card number as stored in the card.

cardExpiryDate is the card's expiry date as stored in the card.

cardInsertionTime is the insertion date and time.

vehicleOdometerValueAtInsertion is the vehicle odometer value at card insertion.

cardSlotNumber is the slot in which the card is inserted.

cardWithdrawalTime is the withdrawal date and time.

vehicleOdometerValueAtWithdrawal is the vehicle odometer value at card withdrawal.

previousVehicleInfo contains information about the previous vehicle used by the driver, as stored in the card.

manualInputFlag is a flag identifying if the cardholder has manually entered driver activities at card insertion.

Generation 2:

Instead of fullCardNumber the generation 2 data structure makes use of the following data element.

fullCardNumberAndGeneration is the type of card, its issuing Member State, its card number and generation as stored in the card.

2.178. **VuCardIWRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to insertion and withdrawal cycles of driver cards or of workshop cards in the vehicle unit (Annex 1C requirement 103).

```
VuCardIWRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF VuCardIWRecord
}
```

recordType denotes the type of the record (VuCardIWRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuCardIWRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of records related to card insertion withdrawal cycles.

[F22.179. **VuCardRecord**

Generation 2:

Information, stored in a vehicle unit, about a tachograph card used (Annex IC requirement 132).

```
VuCardRecord ::= SEQUENCE {
cardNumberAndGenerationInformation      FullCardNumberAndGeneration,
cardExtendedSerialNumber                ExtendedSerialNumber,
cardStructureVersion                    CardStructureVersion,
cardNumber                              CardNumber
}
```

cardNumberAndGenerationInformation is the full card number and generation of the card used (data type 2.74).

cardExtendedSerialNumber as read from the file EF_ICC under the MF of the card.

cardStructureVersion as read from the file EF_Application_Identification under the DF_Tachograph_G2.

cardNumber as read from the file EF_Identification under the DF_Tachograph_G2.**]**

2.180. **VuCardRecordArray**

Generation 2:

Information stored in a vehicle unit about the tachograph cards used with this VU. This information is intended for the analysis of VU — card problems (Annex 1C requirement 132).

```
VuCardRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF VuCardRecord
}
```

recordType denotes the type of the record (VuCardRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuCardRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of records related to the tachograph cards used with the VU.

2.181. **VuCertificate**

Certificate of the public key of a vehicle unit.

```
VuCertificate ::= Certificate
```

2.182. **VuCertificateRecordArray**

Generation 2:

The VU certificate plus metadata as used in the download protocol.

```
VuCertificateRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF VuCertificate
}
```

recordType denotes the type of the record (VuCertificate). **Value Assignment**: See RecordType

recordSize is the size of the VuCertificate in bytes.

noOfRecords is the number of records in the set records. The value shall be set to 1 as the certificates may have different lengths.

records is a set of VU certificates.

2.183. **VuCompanyLocksData**

Generation 1:

Information, stored in a vehicle unit, related to company locks (Annex 1B requirement 104).

```
VuCompanyLocksData ::= SEQUENCE {
   noOfLocks                           INTEGER(0..255),
   vuCompanyLocksRecords               SET SIZE(noOfLocks) OF VuCompanyLocksRecord
}
```

noOfLocks is the number of locks listed in vuCompanyLocksRecords.

vuCompanyLocksRecords is the set of company locks records.

2.184. **VuCompanyLocksRecord**

Information, stored in a vehicle unit, related to one company lock (Annex 1B requirement 104 and Annex 1C requirement 128).

Generation 1:

lockInTime, lockOutTime are the date and time of lock-in and lock-out.

companyName, companyAddress are the company name and address related with the lock-in.

companyCardNumber identifies the card used at lock-in.

Generation 2:

Instead of companyCardNumber the generation 2 data structure makes use of the following data element.

companyCardNumberAndGeneration identifies the card including its generation used at lock-in.

2.185. **VuCompanyLocksRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to company locks (Annex 1C requirement 128).

```
VuCompanyLocksRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       VuCompanyLocksRecord
}
```

recordType denotes the type of the record (VuCompanyLocksRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuCompanyLocksRecord in bytes.

noOfRecords is the number of records in the set records. Value 0..255.

records is the set of company locks records.

2.186. **VuControlActivityData**

Generation 1:

Information, stored in a vehicle unit, related to controls performed using this VU (Annex 1B requirement 102).

```
VuControlActivityData ::= SEQUENCE {
   noOfControls                        INTEGER(0..20),
   vuControlActivityRecords            SET SIZE(noOfControls) OF
                                       VuControlActivityRecord
}
```

noOfControls is the number of controls listed in vuControlActivityRecords.

vuControlActivityRecords is the set of control activity records.

2.187. **VuControlActivityRecord**

Information, stored in a vehicle unit, related to a control performed using this VU (Annex 1B requirement 102 and Annex 1C requirement 126).

Generation 1:

controlType is the type of the control.

controlTime is the date and time of the control.

controlCardNumber identifies the control card used for the control.

downloadPeriodBeginTime is the begin time of the downloaded period, in case of downloading.

downloadPeriodEndTime is the end time of the downloaded period, in case of downloading.

Generation 2:

Instead of controlCardNumber the generation 2 data structure makes use of the following data element.

controlCardNumberAndGeneration identifies the control card including its generation used for the control.

2.188. **VuControlActivityRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to controls performed using this VU (Annex 1C requirement 126).

```
VuControlActivityRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       VuControlActivityRecord
}
```

recordType denotes the type of the record (VuControlActivityRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuControlActivityRecord in bytes.

noOfRecords is the number of records in the set records.

records is the set of VU control activity records.

2.189. **VuDataBlockCounter**

Counter, stored in a card, identifying sequentially the insertion withdrawal cycles of the card in vehicle units.

```
VuDataBlockCounter ::= BCDString(SIZE(2))
```

Value assignment: Consecutive Number with max, value 9 999, starting again with 0.

2.190. **VuDetailedSpeedBlock**

Information, stored in a vehicle unit, related to the vehicle's detailed speed for a minute during which the vehicle has been moving (Annnex 1B requirement 093 and Annex 1C requirement 116).

```
VuDetailedSpeedBlock ::= SEQUENCE {
   speedBlockBeginDate    TimeReal,
   speedsPerSecond        SEQUENCE SIZE(60) OF Speed
}
```

speedBlockBeginDate is the date and time of the first speed value within the block.

speedsPerSecond is the chronological sequence of measured speeds every seconds for the minute starting at speedBlockBeginDate (included).

2.191. **VuDetailedSpeedBlockRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to the detailed speed of the vehicle.

```
VuDetailedSpeedBlockRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       VuDetailedSpeedBlock
}
```

recordType denotes the type of the record (VuDetailedSpeedBlock). **Value Assignment**: See RecordType

recordSize is the size of the VuDetailedSpeedBlock in bytes.

noOfRecords is the number of records in the set records.

records is the set of detailed speed blocks.

2.192. **VuDetailedSpeedData**

Generation 1:

Information, stored in a vehicle unit, related to the detailed speed of the vehicle.

```
VuDetailedSpeedData ::= SEQUENCE {
   noOfSpeedBlocks                     INTEGER(0..2^16-1),
   vuDetailedSpeedBlocks               SET SIZE(noOfSpeedBlocks) OF
                                       VuDetailedSpeedBlock
}
```

noOfSpeedBlocks is the number of speed blocks in the vuDetailedSpeedBlocks set.

vuDetailedSpeedBlocks is the set of detailed speed blocks.

2.193. **VuDownloadablePeriod**

Oldest and latest dates for which a vehicle unit holds data related to drivers activities (Annex 1B requirements 081, 084 or 087 and Annex 1C requirements 102, 105, 108).

```
VuDownloadablePeriod ::= SEQUENCE {
   minDownloadableTime                    TimeReal
   maxDownloadableTime                    TimeReal
}
```

minDownloadableTime is the oldest card insertion or activity change or place entry date and time stored in the VU.

maxDownloadableTime is the latest card withdrawal or activity change or place entry date and time stored in the VU.

2.194. **VuDownloadablePeriodRecordArray**

Generation 2:

The VUDownloadablePeriod plus metadata used in the download protocol.

```
VuDownloadablePeriodRecordArray ::= SEQUENCE {
   recordType                             RecordType,
   recordSize                             INTEGER(1..65535),
   noOfRecords                            INTEGER(0..65535),
   records                                SET SIZE(noOfRecords) OF
                                          VuDownloadablePeriod
}
```

recordType denotes the type of the record (VuDownloadablePeriod). **Value Assignment**: See RecordType

recordSize is the size of the VuDownloadablePeriod in bytes.

noOfRecords is the number of records in the set records.

records is the set of VuDownloadablePeriod records.

2.195. **VuDownloadActivityData**

Information, stored in a vehicle unit, related to its last download (Annex 1B requirement 105 and Annex 1C requirement 129).

Generation 1:

downloadingTime is the date and time of downloading.

fullCardNumber identifies the card used to authorise the download.

companyOrWorkshopName is the company or workshop name.

Generation 2:

Instead of fullCardNumber the generation 2 data structure makes use of the following data element.

fullCardNumberAndGeneration identifies the card including its generation used to authorise the download.

2.196. **VuDownloadActivityDataRecordArray**

Generation 2:

Information related to the last VU download (Annex 1C requirement 129).

```
VuDownloadActivityDataRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF VuDownloadActivityData
}
```

recordType denotes the type of the record (VuDownloadActivityData). **Value Assignment**: See RecordType

recordSize is the size of the VuDownloadActivityData in bytes.

noOfRecords is the number of records in the set records.

records is the set of download activity data records.

2.197. **VuEventData**

Generation 1:

Information, stored in a vehicle unit, related to events (Annex 1B requirement 094 except over speeding event).

```
VuEventData ::= SEQUENCE {
   noOfVuEvents                        INTEGER(0..255),
   vuEventRecords                      SET SIZE(noOfVuEvents) OF VuEventRecord
}
```

noOfVuEvents is the number of events listed in the vuEventRecords set.

vuEventRecords is a set of events records.

2.198. **VuEventRecord**

Information, stored in a vehicle unit, related to an event (Annex 1B requirement 094 and Annex 1C requirement 117 except over speeding event).

Generation 1:

eventType is the type of the event.

eventRecordPurpose is the purpose for which this event has been recorded.

eventBeginTime is the date and time of beginning of event.

eventEndTime is the date and time of end of event.

cardNumberDriverSlotBegin identifies the card inserted in the driver slot at the beginning of the event.

cardNumberCodriverSlotBegin identifies the card inserted in the co-driver slot at the beginning of the event.

cardNumberDriverSlotEnd identifies the card inserted in the driver slot at the end of the event.

cardNumberCodriverSlotEnd identifies the card inserted in the co-driver slot at the end of the event.

similarEventsNumber is the number of similar events that day.

This sequence can be used for all events other than over speeding events.

Generation 2:

In addition to generation 1 the following data elements are used:

manufacturerSpecificEventFaultData contains additional, manufacturer specific information about the event.

Instead of cardNumberDriverSlotBegin, cardNumberCodriverSlotBegin, cardNumberDriverSlotEnd, and cardNumberCodriverSlotEnd the generation 2 data structure makes use of the following data elements:

cardNumberAndGenDriverSlotBegin identifies the card including its generation which is inserted in the driver slot at the beginning of the event.

cardNumberAndGenCodriverSlotBegin identifies the card including its generation which is inserted in the co-driver slot at the beginning of the event.

cardNumberAndGenDriverSlotEnd identifies the card including its generation which is inserted in the driver slot at the end of the event.

cardNumberAndGenCodriverSlotEnd identifies the card including its generation which is inserted in the co-driver slot at the end of the event.

If the event is a time conflict the eventBeginTime and eventEndTime are to be interpreted as follows:

eventBeginTime is the recording equipment date and time.

eventEndTime is the GNSS date and time.

2.199. **VuEventRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to events (Annex 1C requirement 117 except over speeding event).

```
VuEventRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF VuEventRecord
}
```

recordType denotes the type of the record (VuEventRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuEventRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of events records.

2.200. **VuFaultData**

Generation 1:

Information, stored in a vehicle unit, related to faults (Annex 1B requirement 096).

```
VuFaultData ::= SEQUENCE {
   noOfVuFaults                        INTEGER(0..255),
   vuFaultRecords                      SET SIZE(noOfVuFaults) OF VuFaultRecord
}
```

noOfVuFaults is the number of faults listed in the vuFaultRecords set.

vuFaultRecords is a set of faults records.

2.201. **VuFaultRecord**

Information, stored in a vehicle unit, related to a fault (Annex 1B requirement 096 and Annex 1C requirement 118).

Generation 1:

faultType is the type of recording equipment fault.

faultRecordPurpose is the purpose for which this fault has been recorded.

faultBeginTime is the date and time of beginning of fault.

faultEndTime is the date and time of end of fault.

cardNumberDriverSlotBegin identifies the card inserted in the driver slot at the beginning of the fault.

cardNumberCodriverSlotBegin identifies the card inserted in the co-driver slot at the beginning of the fault.

cardNumberDriverSlotEnd identifies the card inserted in the driver slot at the end of the fault.

cardNumberCodriverSlotEnd identifies the card inserted in the co-driver slot at the end of the fault.

Generation 2:

In addition to generation 1 the following data element is used:

manufacturerSpecificEventFaultData contains additional, manufacturer specific information about the fault.

Instead of cardNumberDriverSlotBegin, cardNumberCodriverSlotBegin, cardNumberDriverSlotEnd, and cardNumberCodriverSlotEnd the generation 2 data structure makes use of the following data elements:

cardNumberAndGenDriverSlotBegin identifies the card including its generation which is inserted in the driver slot at the beginning of the fault.

cardNumberAndGenCodriverSlotBegin identifies the card including its generation which is inserted in the co-driver slot at the beginning of the fault.

cardNumberAndGenDriverSlotEnd identifies the card including its generation which is inserted in the driver slot at the end of the fault.

cardNumberAndGenCodriverSlotEnd identifies the card including its generation which is inserted in the co-driver slot at the end of the fault.

2.202. **VuFaultRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to faults (Annex 1C requirement 118).

```
VuFaultRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF VuFaultRecord
}
```

recordType denotes the type of the record (VuFaultRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuFaultRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of faults records.

[F2 2.203. **VuGNSSADRecord**

Generation 2:

Information, stored in a vehicle unit, related to the GNSS position of the vehicle if the accumulated driving time reaches a multiple of three hours (Annex IC requirement 108, 110).

```
VuGNSSADRecord ::= SEQUENCE {
   timeStamp                        TimeReal,
   cardNumberAndGenDriverSlot       FullCardNumberAndGeneration,
   cardNumberAndGenCodriverSlot     FullCardNumberAndGeneration,
   gnssPlaceRecord                  GNSSPlaceRecord,
   vehicleOdometerValue             OdometerShort
}
```

timeStamp is the date and time when the accumulated driving time reaches a multiple of three hours.

cardNumberAndGenDriverSlot identifies the card including its generation which is inserted in the driver slot.

cardNumberAndGenCodriverSlot identifies the card including its generation which is inserted in the co-driver slot.

gnssPlaceRecord contains information related to the position of the vehicle.

vehicleOdometerValue is the odometer value when the accumulated driving time reaches a multiple of three hours.

2.204. **VuGNSSADRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to the GNSS position of the vehicle if the accumulated driving time reaches a multiple of three hours (Annex IC requirement 108 and 110).

```
VuGNSSADRecordArray ::= SEQUENCE {
   recordType                   RecordType,
   recordSize                   INTEGER(1..65535),
   noOfRecords                  INTEGER(0..65535),
   records                      SET SIZE(noOfRecords) OF VuGNSSADRecord
}
```

recordType denotes the type of the record (VuGNSSADRecord).

Value Assignment: See RecordType.

recordSize is the size of the VuGNSSADRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of GNSS accumulated driving records.**]**

2.205. **VuIdentification**

Information, stored in a vehicle unit, related to the identification of the vehicle unit (Annex 1B requirement 075 and Annex 1C requirement 93 and 121).

Generation 1:

vuManufacturerName is the name of the manufacturer of the vehicle unit.

vuManufacturerAddress is the address of the manufacturer of the vehicle unit.

vuPartNumber is the part number of the vehicle unit.

vuSerialNumber is the serial number of the vehicle unit.

vuSoftwareIdentification identifies the software implemented in the vehicle unit.

vuManufacturingDate is the manufacturing date of the vehicle unit.

vuApprovalNumber is the type approval number of the vehicle unit.

Generation 2:

In addition to generation 1 the following data element are used:

vuGeneration identifies the generation of the vehicle unit.

vuAbility provides information whether the VU supports generation 1 tachograph cards or not.

2.206. **VuIdentificationRecordArray**

Generation 2:

The VuIdentification plus metadata used in the download protocol.

```
VuIdentificationRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF VuIdentification
}
```

recordType denotes the type of the record (VuIdentification). **Value Assignment**: See RecordType

recordSize is the size of the VuIdentification in bytes.

noOfRecords is the number of records in the set records.

records is a set of VuIdentification records.

2.207. **VuITSConsentRecord**

Generation 2:

Information stored in a vehicle unit, related to the consent of a driver to use Intelligent Transport Systems.

```
VuITSConsentRecord ::= SEQUENCE {
   cardNumberAndGen                    FullCardNumberAndGeneration,
   consent                             BOOLEAN
}
```

cardNumberAndGen identifies the card including its generation. This must be a driver card or a workshop card.

consent is a flag which indicates whether the driver has given his consent on the usage of Intelligent Transport Systems with this vehicle / vehicle unit.

Value assignment:

TRUE	indicates the driver's consent to use Intelligent Transport Systems
FALSE	indicates the driver's denial to use Intelligent Transport Systems

2.208. **VuITSConsentRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to drivers' consent on the usage of Intelligent Transport Systems (Annex 1C requirement 200).

```
VuITSConsentRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF VuITSConsentRecord
}
```

recordType denotes the type of the record (VuITSConsentRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuITSConsentRecord in bytes.

noOfRecords is the number of records in the set records.

records is the set of ITS consent records.

2.209. **VuManufacturerAddress**

Address of the manufacturer of the vehicle unit.

```
VuManufacturerAddress ::= Address
```

Value assignment: Unspecified.

2.210. **VuManufacturerName**

Name of the manufacturer of the vehicle unit.

```
VuManufacturerName ::= Name
```

Value assignment: Unspecified.

2.211. **VuManufacturingDate**

Date of manufacture of the vehicle unit.

```
VuManufacturingDate ::= TimeReal
```

Value assignment: Unspecified.

2.212. **VuOverSpeedingControlData**

Information, stored in a vehicle unit, related to over speeding events since the last over speeding control (Annex 1B requirement 095 and Annex 1C requirement 117).

```
VuOverSpeedingControlData ::= SEQUENCE {
   lastOverspeedControlTime              TimeReal,
   firstOverspeedSince                   TimeReal,
   numberOfOverspeedSince                OverspeedNumber
}
```

lastOverspeedControlTime is the date and time of the last over speeding control.

firstOverspeedSince is the date and time of the first over speeding following this over speeding control.

numberOfOverspeedSince is the number of over speeding events since the last over speeding control.

2.213. **VuOverSpeedingControlDataRecordArray**

Generation 2:

The VuOverSpeedingControlData plus metadata used in the download protocol.

```
VuOverSpeedingControlDataRecordArray ::= SEQUENCE {
   recordType                            RecordType,
   recordSize                            INTEGER(1..65535),
   noOfRecords                           INTEGER(0..65535),
   records                               SET SIZE(noOfRecords) OF
                                         VuOverSpeedingControlData
}
```

recordType denotes the type of the record (VuOverSpeedingControlData). **Value Assignment**: See RecordType

recordSize is the size of the VuOverSpeedingControlData in bytes.

noOfRecords is the number of records in the set records.

records is a set of over speeding control data records.

2.214. **VuOverSpeedingEventData**

Generation 1:

Information, stored in a vehicle unit, related to over speeding events (Annex 1B requirement 094).

```
VuOverSpeedingEventData ::= SEQUENCE {
   noOfVuOverSpeedingEvents              INTEGER(0..255),
   vuOverSpeedingEventRecords            SET SIZE(noOfVuOverSpeedingEvents) OF
                                         VuOverSpeedingEventRecord
}
```

noOfVuOverSpeedingEvents is the number of events listed in the vuOverSpeedingEventRecords set.

vuOverSpeedingEventRecords is a set of over speeding events records.

2.215. **VuOverSpeedingEventRecord**

Generation 1:

Information, stored in a vehicle unit, related to over speeding events (Annex 1B requirement 094 and Annex 1C requirement 117).

eventType is the type of the event.

eventRecordPurpose is the purpose for which this event has been recorded.

eventBeginTime is the date and time of beginning of event.

eventEndTime is the date and time of end of event.

maxSpeedValue is the maximum speed measured during the event.

averageSpeedValue is the arithmetic average speed measured during the event.

cardNumberDriverSlotBegin identifies the card inserted in the driver slot at the beginning of the event.

similarEventsNumber is the number of similar events that day.

Generation 2:

Information, stored in a vehicle unit, related to over speeding events (Annex 1B requirement 094 and Annex 1C requirement 117).

Instead of cardNumberDriverSlotBegin, the generation 2 data structure makes use of the following data element:

cardNumberAndGenDriverSlotBegin identifies the card including its generation which is inserted in the driver slot at the beginning of the event.

2.216. VuOverSpeedingEventRecordArray

Generation 2:

Information, stored in a vehicle unit, related to over speeding events (Annex 1C requirement 117).

```
VuOverSpeedingEventRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       VuOverSpeedingEventRecord
}
```

recordType denotes the type of the record (VuOverSpeedingEventRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuOverSpeedingEventRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of over speeding events records.

2.217. VuPartNumber

Part number of the vehicle unit.

```
VuPartNumber ::= IA5String(SIZE(16))
```

Value assignment: VU manufacturer specific.

2.218. **VuPlaceDailyWorkPeriodData**

Generation 1:

Information, stored in a vehicle unit, related to places where drivers begin or end a daily work period (Annex 1B requirement 087 and Annex 1C requirement 108 and 110).

```
VuPlaceDailyWorkPeriodData ::= SEQUENCE {
   noOfPlaceRecords                    INTEGER(0..255),
   vuPlaceDailyWorkPeriodRecords       SET SIZE(noOfPlaceRecords) OF
                                       VuPlaceDailyWorkPeriodRecord
}
```

noOfPlaceRecords is the number of records listed in the vuPlaceDailyWorkPeriodRecords set.

vuPlaceDailyWorkPeriodRecords is a set of place related records.

2.219. **VuPlaceDailyWorkPeriodRecord**

Generation 1:

Information, stored in a vehicle unit, related to a place where a driver begins or ends a daily work period (Annex 1B requirement 087 and Annex 1C requirement 108 and 110).

fullCardNumber is the driver's card type, card issuing Member State and card number.

placeRecord contains the information related to the place entered.

Generation 2:

Information, stored in a vehicle unit, related to a place where a driver begins or ends a daily work period (Annex 1B requirement 087 and Annex 1C requirement 108 and 110).

Instead of fullCardNumber, the generation 2 data structure makes use of the following data element:

fullCardNumberAndGeneration is the type of card, its issuing Member State, its card number and generation as stored in the card.

2.220. **VuPlaceDailyWorkPeriodRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to places where drivers begin or end a daily work period (Annex 1C requirement 108 and 110).

```
VuPlaceDailyWorkPeriodRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       VuPlaceDailyWorkPeriodRecord
}
```

recordType denotes the type of the record (VuPlaceDailyWorkPeriodRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuPlaceDailyWorkPeriodRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of place related records.

2.221. **VuPrivateKey**

Generation 1:

The private key of a vehicle unit.

```
VuPrivateKey ::= RSAKeyPrivateExponent
```

2.222. **VuPublicKey**

Generation 1:

The public key of a vehicle unit.

```
VuPublicKey ::= PublicKey
```

2.223. **VuSerialNumber**

Serial number of the vehicle unit (Annex 1B requirement 075 and Annex 1C requirement 93).

```
VuSerialNumber ::= ExtendedSerialNumber
```

2.224. **VuSoftInstallationDate**

Date of installation of the vehicle unit software version.

```
VuSoftInstallationDate ::= TimeReal
```

Value assignment: Unspecified.

2.225. **VuSoftwareIdentification**

Information, stored in a vehicle unit, related to the software installed.

```
VuSoftwareIdentification ::= SEQUENCE {
   vuSoftwareVersion                VuSoftwareVersion,
   vuSoftInstallationDate           VuSoftInstallationDate
}
```

vuSoftwareVersion is the software version number of the Vehicle Unit.

vuSoftInstallationDate is the software version installation date.

2.226. **VuSoftwareVersion**

Software version number of the vehicle unit.

```
VuSoftwareVersion ::= IA5String(SIZE(4))
```

Value assignment: Unspecified.

2.227. **VuSpecificConditionData**

Generation 1:

Information, stored in a vehicle unit, related to specific conditions.

```
VuSpecificConditionData ::= SEQUENCE {
   noOfSpecificConditionRecords         INTEGER(0..2^16-1)
   specificConditionRecords             SET SIZE (noOfSpecificConditionRecords) OF
                                        SpecificConditionRecord
}
```

noOfSpecificConditionRecords is the number of records listed in the specificConditionRecords set.

specificConditionRecords is a set of specific conditions related records.

2.228. **VuSpecificConditionRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to specific conditions (Annex 1C requirement 130).

```
VuSpecificConditionRecordArray ::= SEQUENCE {
   recordType                           RecordType,
   recordSize                           INTEGER(1..65535),
   noOfRecords                          INTEGER(0..65535),
   records                              SET SIZE(noOfRecords) OF
                                        SpecificConditionRecord
}
```

recordType denotes the type of the record (SpecificConditionRecord). **Value Assignment**: See RecordType

recordSize is the size of the SpecificConditionRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of specific conditions related records.

2.229. **VuTimeAdjustmentData**

Generation 1:

Information, stored in a vehicle unit, related to time adjustments performed outside the frame of a regular calibration (Annex 1B requirement 101).

```
VuTimeAdjustmentData ::= SEQUENCE {
   noOfVuTimeAdjRecords                 INTEGER(0..6),
   vuTimeAdjustmentRecords              SET SIZE(noOfVuTimeAdjRecords) OF
                                        VuTimeAdjustmentRecord
}
```

noOfVuTimeAdjRecords is the number of records in vuTimeAdjustmentRecords.

vuTimeAdjustmentRecords is a set of time adjustment records.

[[F2]2.230. **Reserved for future use**

2.231. **Reserved for future use]**

2.232. **VuTimeAdjustmentRecord**

Information, stored in a vehicle unit, related a time adjustment performed outside the frame of a regular calibration (Annex 1B requirement 101 and Annex 1C requirement 124 and 125).

Generation 1:

oldTimeValue, newTimeValue are the old and new values of date and time.

workshopName, workshopAddress are the workshop name and address.

workshopCardNumber identifies the workshop card used to perform the time adjustment.
Generation 2:

Instead of workshopCardNumber the generation 2 data structure makes use of the following data element.

workshopCardNumberAndGeneration identifies the workshop card including its generation used to perform the time adjustment.

2.233. **VuTimeAdjustmentRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to time adjustments performed outside the frame of a regular calibration (Annex 1C requirement 124 and 125).

```
VuTimeAdjustmentRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       VuTimeAdjustmentRecord
}
```

recordType denotes the type of the record (VuTimeAdjustmentRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuTimeAdjustmentRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of time adjustment records.

2.234. **WorkshopCardApplicationIdentification**

Information, stored in a workshop card related to the identification of the application of the card (Annex 1C requirement 307 and 330).
Generation 1:

typeOfTachographCardId is specifying the implemented type of card.

cardStructureVersion is specifying the the version of the structure that is implemented in the card.

noOfEventsPerType is the number of events per type of event the card can record.

noOfFaultsPerType is the number of faults per type of fault the card can record.

activityStructureLength indicates the number of bytes available for storing activity records.

noOfCardVehicleRecords is the number of vehicle records the card can contain.

noOfCardPlaceRecords is the number of places the card can record.

noOfCalibrationRecords is the number of calibration records the card can store.

Generation 2:

[[F2]In addition to generation 1 the following data elements are used:

noOfGNSSADRecords is the number of GNSS accumulated driving records the card can store.

noOfSpecificConditionRecords is the number of specific condition records the card can store.

noOfCardVehicleUnitRecords is the number of vehicle units used records the card can store.**]**

2.235. **WorkshopCardCalibrationData**

Information, stored in a workshop card, related to workshop activity performed with the card (Annex 1C requirements 314, 316, 337, and 339).

```
WorkshopCardCalibrationData ::= SEQUENCE {
   calibrationTotalNumber              INTEGER(0 .. 2^16-1),
   calibrationPointerNewestRecord      INTEGER(0 .. NoOfCalibrationRecords-1),
   calibrationRecords                  SET SIZE(NoOfCalibrationRecords) OF
                                       WorkshopCardCalibrationRecord
}
```

calibrationTotalNumber is the total number of calibrations performed with the card.

calibrationPointerNewestRecord is the index of the last updated calibration record.

Value assignment: Number corresponding to the numerator of the calibration record, beginning with '0' for the first occurrence of the calibration records in the structure.

calibrationRecords is the set of records containing calibration and/or time adjustment information.

2.236. **WorkshopCardCalibrationRecord**

Information, stored in a workshop card, related to a calibration performed with the card (Annex 1C requirement 314 and 337).

Generation 1:

calibrationPurpose is the purpose of the calibration.

vehicleIdentificationNumber is the VIN.

vehicleRegistration contains the VRN and registering Member State.

wVehicleCharacteristicConstant is the characteristic coefficient of the vehicle.

kConstantOfRecordingEquipment is the constant of the recording equipment.

lTyreCircumference is the effective circumference of the wheel tyres.

tyreSize is the designation of the dimensions of the tyres mounted on the vehicle.

authorisedSpeed is the maximum authorised speed of the vehicle.

oldOdometerValue, newOdometerValue are the old and new values of the odometer.

oldTimeValue, newTimeValue are the old and new values of date and time.

nextCalibrationDate is the date of the next calibration of the type specified in CalibrationPurpose to be carried out by the authorised inspection authority.

vuPartNumber, vuSerialNumber and **sensorSerialNumber** are the data elements for recording equipment identification.
Generation 2:

In addition to generation 1 the following data elements are used:

sensorGNSSSerialNumber which identifies an external GNSS facility.

rcmSerialNumber which identifies a Remote Communication Module.

sealDataCard gives information about the seals that are attached to different components of the vehicle.

2.237. **WorkshopCardHolderIdentification**

Information, stored in a workshop card, related to the identification of the cardholder (Annex 1C requirement 311 and 334).

```
WorkshopCardHolderIdentification ::= SEQUENCE {
   workshopName                        Name,
   workshopAddress                     Address,
   cardHolderName                      HolderName,
   cardHolderPreferredLanguage         Language
}
```

workshopName is name of the workshop of the card holder.

workshopAddress is the address of the workshop of the card holder.

cardHolderName is the name and first name(s) of the holder (e.g. the name of the mechanic).

cardHolderPreferredLanguage is the preferred language of the card holder.

2.238. **WorkshopCardPIN**

Personal identification number of the Workshop Card (Annex 1C requirement 309 and 332).

```
WorkshopCardPIN ::= IA5String(SIZE(8))
```

Value assignment: The PIN known to the cardholder, right padded with 'FF' bytes up to 8 bytes.

2.239. **W-VehicleCharacteristicConstant**

Characteristic coefficient of the vehicle (definition k)).

```
W-VehicleCharacteristicConstant ::= INTEGER(0..2^16-1))
```

Value assignment: Impulses per kilometer in the operating range 0 to 64 255 pulses/km.

2.240. **VuPowerSupplyInterruptionRecord**

Generation 2:

Information, stored in a vehicle unit, related to Power Supply Interruption events (Annex 1C requirement 117).

```
VuPowerSupplyInterruptionRecord ::= SEQUENCE {
   eventType                          EventFaultType,
   eventRecordPurpose                 EventFaultRecordPurpose,
   eventBeginTime                     TimeReal,
   eventEndTime                       TimeReal,
   cardNumberAndGenDriverSlotBegin    FullCardNumberAndGeneration,
   cardNumberAndGenDriverSlotEnd      FullCardNumberAndGeneration,
   cardNumberAndGenCodriverSlotBegin  FullCardNumberAndGeneration,
   cardNumberAndGenCodriverSlotEnd    FullCardNumberAndGeneration,
   similarEventsNumber                SimilarEventsNumber
}
```

eventType is the type of the event.

eventRecordPurpose is the purpose for which this event has been recorded.

eventBeginTime is the date and time of beginning of event.

eventEndTime is the date and time of end of event.

cardNumberAndGenDriverSlotBegin identifies the card including its generation inserted in the driver slot at the beginning of the event.

cardNumberAndGenDriverSlotEnd identifies the card including its generation inserted in the driver slot at the end of the event.

cardNumberAndGenCodriverSlotBegin identifies the card including its generation inserted in the co-driver slot at the beginning of the event.

cardNumberAndGenCodriverSlotEnd identifies the card including its generation inserted in the co-driver slot at the end of the event.

similarEventsNumber is the number of similar events that day.

2.241. **VuPowerSupplyInterruptionRecordArray**

Generation 2:

Information, stored in a vehicle unit, related to Power Supply Interruption events (Annex 1C requirement 117).

```
VuPowerSupplyInterruptionRecordArray ::= SEQUENCE {
   recordType                         RecordType,
   recordSize                         INTEGER(1..65535),
   noOfRecords                        INTEGER(0..65535),
   records                            SET SIZE(noOfRecords) OF
                                      VuPowerSupplyInterruptionRecord
}
```

recordType denotes the type of the record (VuPowerSupplyInterruptionRecord). **Value Assignment**: See RecordType

recordSize is the size of the VuPowerSupplyInterruptionRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of power supply interruption events records.

2.242. **VuSensorExternalGNSSCoupledRecordArray**

Generation 2:

A set of SensorExternalGNSSCoupledRecord plus metadata used in the download protocol.

```
VuSensorExternalGNSSCoupledRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF
                                       SensorExternalGNSSCoupledRecord
}
```

recordType denotes the type of the record (SensorExternalGNSSCoupledRecord). **Value Assignment**: See RecordType

recordSize is the size of the SensorExternalGNSSCoupledRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of Sensor External GNSS Coupled records.

2.243. **VuSensorPairedRecordArray**

Generation 2:

A set of SensorPairedRecord plus metadata used in the download protocol.

```
VuSensorPairedRecordArray ::= SEQUENCE {
   recordType                          RecordType,
   recordSize                          INTEGER(1..65535),
   noOfRecords                         INTEGER(0..65535),
   records                             SET SIZE(noOfRecords) OF SensorPairedRecord
}
```

recordType denotes the type of the record (SensorPairedRecord). **Value Assignment**: See RecordType

recordSize is the size of the SensorPairedRecord in bytes.

noOfRecords is the number of records in the set records.

records is a set of sensor paired records.

3. VALUE AND SIZE RANGE DEFINITIONS

Definition of variable values used for definitions in paragraph 2.

```
TimeRealRange ::= 2^32-1
```

4. CHARACTER SETS

IA5Strings use the ASCII characters as defined by ISO/IEC 8824-1. For readability and for easy referencing the value assignment is given below. The ISO/IEC 8824-1 supersedes this informative note in case of discrepancy.

```
  ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
```

Other character strings (Address, Name, VehicleRegistrationNumber) use, in addition, characters from the decimal character code range 161 —	**Code Page(Decimal)**

255 of the following 8-bit, standard character sets, specified by the Code Page number:Standard Character Set	
ISO/IEC 8859-1 Latin-1 Western European	1
ISO/IEC 8859-2 Latin-2 Central European	2
ISO/IEC 8859-3 Latin-3 South European	3
ISO/IEC 8859-5 Latin / Cyrillic	5
ISO/IEC 8859-7 Latin / Greek	7
ISO/IEC 8859-9 Latin-5 Turkish	9
ISO/IEC 8859-13 Latin-7 Baltic Rim	13
ISO/IEC 8859-15 Latin-9	15
ISO/IEC 8859-16 Latin-10 South Eastern European	16
KOI8-R Latin / Cyrillic	80
KOI8-U Latin / Cyrillic	85

5. ENCODING

When encoded with ASN.1 encoding rules, all data types defined shall be encoded according to ISO/IEC 8825-2, aligned variant.

6. OBJECT IDENTIFIERS UND APPLICATION IDENTIFIERS

6.1. **Object Identifiers**

The Object Identifiers (OIDs) listed in this chapter are only relevant for generation 2. These OIDs are specified in TR-03110-3 and repeated here for the sake of completeness. These OIDs are contained in the subtree of bsi-de:

```
bsi-de OBJECT IDENTIFIER ::= {
   itu-t(0) identified-organization(4) etsi(0)
   reserved(127) etsi-identified-organization(0) 7
}
```

VU Authentication protocol identifiers

```
id-TA         OBJECT IDENTIFIER ::= {bsi-de protocols(2) smartcard(2) 2}

id-TA-ECDSA                OBJECT IDENTIFIER ::= {id-TA 2}

id-TA-ECDSA-SHA-256 OBJECT IDENTIFIER ::= {id-TA-ECDSA 3}

id-TA-ECDSA-SHA-384 OBJECT IDENTIFIER ::= {id-TA-ECDSA 4}

id-TA-ECDSA-SHA-512 OBJECT IDENTIFIER ::= {id-TA-ECDSA 5}
```

Example: Suppose VU Authentication is to be done with SHA-384, then the object identifier to use is (in ASN.1 notation) `bsi-de protocols(2) smartcard(2) 2 2 4`. The value of this object identifier in dot notation is `0.4.0.127.0.7.2.2.2.2.4`.

	Dot notation	**Byte notation**

id-TA-ECDSA-SHA-256	0.4.0.127.0.7.2.2.2.2.3	‘04 00 7F 00 07 02 02 02 02 03’
id-TA-ECDSA-SHA-384	0.4.0.127.0.7.2.2.2.2.4	‘04 00 7F 00 07 02 02 02 02 04’
id-TA-ECDSA-SHA-512	0.4.0.127.0.7.2.2.2.2.5	‘04 00 7F 00 07 02 02 02 02 05’

Chip Authentication protocol identifiers

```
id-CA          OBJECT IDENTIFIER ::= {bsi-de protocols(2) smartcard(2) 3}

id-CA-ECDH                          OBJECT IDENTIFIER ::= {id-CA 2}

id-CA-ECDH-AES-CBC-CMAC-128         OBJECT IDENTIFIER ::= {id-CA-ECDH 2}

id-CA-ECDH-AES-CBC-CMAC-192         OBJECT IDENTIFIER ::= {id-CA-ECDH 3}

id-CA-ECDH-AES-CBC-CMAC-256         OBJECT IDENTIFIER ::= {id-CA-ECDH 4}
```

Example: Suppose Chip Authentication is to be done by using the ECDH algorithm, resulting in an AES session key length of 128 bits. This session key will subsequently be used in the CBC mode of operation to ensure data confidentiality and with the CMAC algorithm to ensure data authenticity. Therefore, the object identifier to use is (in ASN.1 notation) `bsi-de protocols(2) smartcard(2) 3 2 2`. The value of this object identifier in dot notation is `0.4.0.127.0.7.2.2.3.2.2`.

	Dot notation	Byte notation
id-CA-ECDH-AES-CBC-CMAC-128	0.4.0.127.0.7.2.2.3.2.2	‘04 00 7F 00 07 02 02 03 02 02’
id-CA-ECDH-AES-CBC-CMAC-192	0.4.0.127.0.7.2.2.3.2.3	‘04 00 7F 00 07 02 02 03 02 03’
id-CA-ECDH-AES-CBC-CMAC-256	0.4.0.127.0.7.2.2.3.2.4	‘04 00 7F 00 07 02 02 03 02 04’

6.2. Application Identifiers

Generation 2:

The Application Identifier (AID) for the External GNSS Facility (Generation 2) is given by ‘FF 44 54 45 47 4D’. This is a proprietary AID according to ISO/IEC 7816-4.

Note: The last 5 bytes encode DTEGM for smart Tachograph External GNSS Facility.

The Application Identifier for the generation 2 tachograph card application is given by ‘FF 53 4D 52 44 54’. This is a proprietary AID according to ISO/IEC 7816-4.

Changes to legislation:
There are outstanding changes not yet made to Commission Implementing Regulation (EU) 2016/799. Any changes that have already been made to the legislation appear in the content and are referenced with annotations.
View outstanding changes

Changes and effects yet to be applied to the whole legislation item and associated provisions

- Signature words omitted by S.I. 2019/453 reg. 110
- Annex 1C modified by S.I. 2023/739 reg. 3Sch.