ASAM MCD-2D (ODX)
Version 2.1.0

Data Model Specification

Association for **S**tandardisation of
Automation and **M**easuring Systems

Dated:15.09.06

Status of Document

Date:	15.09.06
Author:	Dr. Augustin, T-Systems Mr. Backmeister, DaimlerChrysler Dr. Beiter, Vector Informatik Mrs. Dogan, Bosch Dr. Hallermayer, BMW Mr. Hecker, Softing Mr. Hümpfner, ESG Mr. Köhler, T-Systems Dr. Kricke, ETAS Mr. Kolbe, Porsche Mr. Michard, Renault Mr. Öhlenschläger, General Motors Mr. Ramrath, In2Soft Dr. Schleicher, DSA Mr. Wallschläger, Audi Mr. Watzal, Siemens VDO Mr. Wolter, Gedas Mr. Zweigler, Siemens
Version:	Version 2.1.0
Doc-ID:	
Status:	Release
Type	ASAM MCD-2D (ODX)

Revision History

This revision history shows only major modifications between release versions.

Date	Author	Filename	Comments
15.09.06	Dr. Augustin, Mr. Hümpfner	ASAM_MCD2D_ODX _V2.1.0.doc	Release Version 2.1.0

Table of contents

Introduction

This International Standard has been established in order to define the requirements of transferring ECU (Electronic Control Unit) diagnostic and programming data between system supplier, vehicle manufacturer and service dealerships.
Today's situation in the automotive industry mostly utilizes paper documentation to document diagnostic data stream information of a vehicle ECU. Each user utilizing the ECU diagnostic data stream documentation to setup development tools or service diagnostic test equipment needs to enter the data documentation into these tools. This redundant effort may no longer be required if those tools support the ODX concept.
This part of ODX includes the data model definition of an ECU diagnostic and programming data in UML (Unified Modelling Language). The document also includes an implementation in XML in the Annex.

1 SCOPE

This part of Road vehicles – ODX (Open diagnostic data exchange) specifies the concept of utilizing a new industry standard diagnostic format to make diagnostic data stream information available to diagnostic tool application manufacturer to simplify the support of the independent aftermarket automotive service industry. The ODX modelled diagnostic data are compatible to the software requirements of the Modular Vehicle Communication Interface (MVCI). This document will enable a MVCI to communicate with the vehicle (ECU(s)) and interpret the diagnostic data contained in the messages exchanged between the external test equipment and the ECU(s). No software programming is necessary to convert diagnostic data into technician readable physical units and text description to be displayed by the tester.
The ODX specification contains the data model to describe all diagnostic data of a vehicle and physical ECU (e.g. diagnostic trouble codes, data parameters, identification data, input/output parameters, variant coding data, communication parameters, etc.). ODX is described in UML (Unified Modelling Language) diagrams and the data exchange format utilizes XML (extensible markup language).
The ODX modelled diagnostic data describe two (2) diagnostic relevant parts of the vehicle:

a) Diagnostic data contained in the ECU, and

b) Diagnostic relevant data required from the vehicle's point of view.

The figure below shows the "central source" origin of diagnostic data, a verification and feedback mechanism with distribution to end-users. Engineering, manufacturing, and service specify which communication protocol and data shall be implemented in the ECU. This information will be documented in a structured format utilizing the XML standard and an appropriate ODX – XML editor. There is potential to develop a concept to generate software source code from the XML file containing diagnostic data. Furthermore, the same XML file is used to setup the diagnostic engineering tools to verify proper communication with the ECU and to perform functional verification and compliance testing. Once all quality goals are met the XML file may be released to an OEM database. Diagnostic information is now available to manufacturing, service, OEM franchised dealers, and aftermarket service outlets via Intranet and Internet.

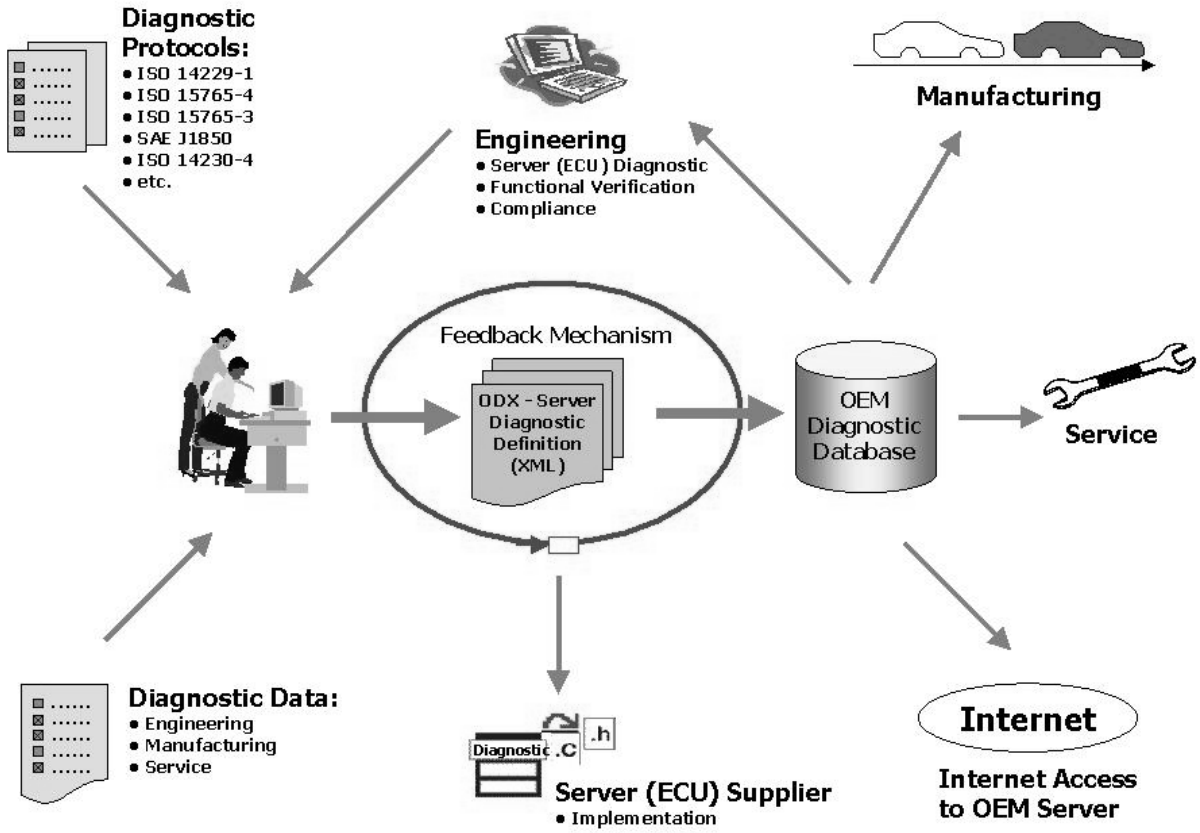


Figure 1 — ODX central source diagnostic data process

The objective of this specification is to ensure that diagnostic data from any vehicle manufacturer is independent of the testing hardware and protocol software supplied by any test equipment manufacturer.

2 NORMATIVE REFERENCES

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 11898-1 (all parts), Road vehicles — Controller Area Network (CAN)

ISO 14229-1, *Road vehicles - Unified diagnostic services — Part 1: Specification and requirements*

ISO 14230 (all parts), *Road vehicles - Diagnostic systems - Keyword protocol 2000*

ISO 15031-5, *Road vehicles - Communication between vehicle and external equipment for emissions-related diagnostics - Part 5: Emissions-related diagnostic services*

ISO 15765 (all parts), *Road vehicles - Diagnostics on controller area network (CAN)*

ISO 9141-2:1994, *Road vehicles - Diagnostic systems - Part 2: CARB requirements for interchange of digital information*

ISO 9141-2:1994/ *Amd.1:1996* *Road vehicles - Diagnostic systems - Part 2: CARB requirements for interchange of digital information Amendment 1*

ISO/IEC 8859-1:1998 *Information technology -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1*

ISO/IEC 8859-2:1999 *Information technology -- 8-bit single-byte coded graphic character sets -- Part 2: Latin alphabet No. 2*

ISO/IEC 10646:2003 *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*

ASAM MCD 2 *Harmonized Data Objects Version 1.0*

3 SYMBOLS AND ABBREVIATED TERMS

API	Application Programming Interface
ASAM	Association for Standardisation of Automation and Measuring Systems
MCD	Measurement, Calibration and Diagnosis
ASCII	American Standard for Character Information Interchange
CRC	Cyclic Redundancy Check
DOP	Diagnostic Data Object Property
ECU	Electronic Control Unit
GMT	Greenwich Mean Time
ODX	Open Diagnostic Data Exchange
OEM	Original Equipment Manufacturer
PC	Personal Computer
PDU	Protocol Data Unit
PDX	Packaged ODX
UC	Use Case
UML	Unified Modelling Language
UTC	Coordinated Universal Time
W3C	World Wide Web Consortium
XML	Extensible Mark-up Language

4 ODX USE CASES

4.1 USE CASE 1: ODX PROCESS CHAIN

Figure 2 shows an example of how ODX data is used within a process chain, which consists of three (3) phases:

a) The phase A development process takes place between vehicle manufacturer and system supplier engineering departments. Both entities exchange ODX data to support the development of the diagnostic implementation in the ECU and the development tools.

b) The phase B development process takes place at the vehicle manufacturer. The engineering departments release the ODX data into an ODX database. The manufacturing and service department use the ODX data as the basis to setup the End-Of-Line test equipment and service application development tools to develop their test application software.

c) The phase C of the development process supports the service dealership diagnostic and programming tools. The service department develops service tool application software, which is based on the released ODX data. The diagnostic and programming software is now available to all service dealerships.

The ODX database is the "central source origin of diagnostic and programming data".

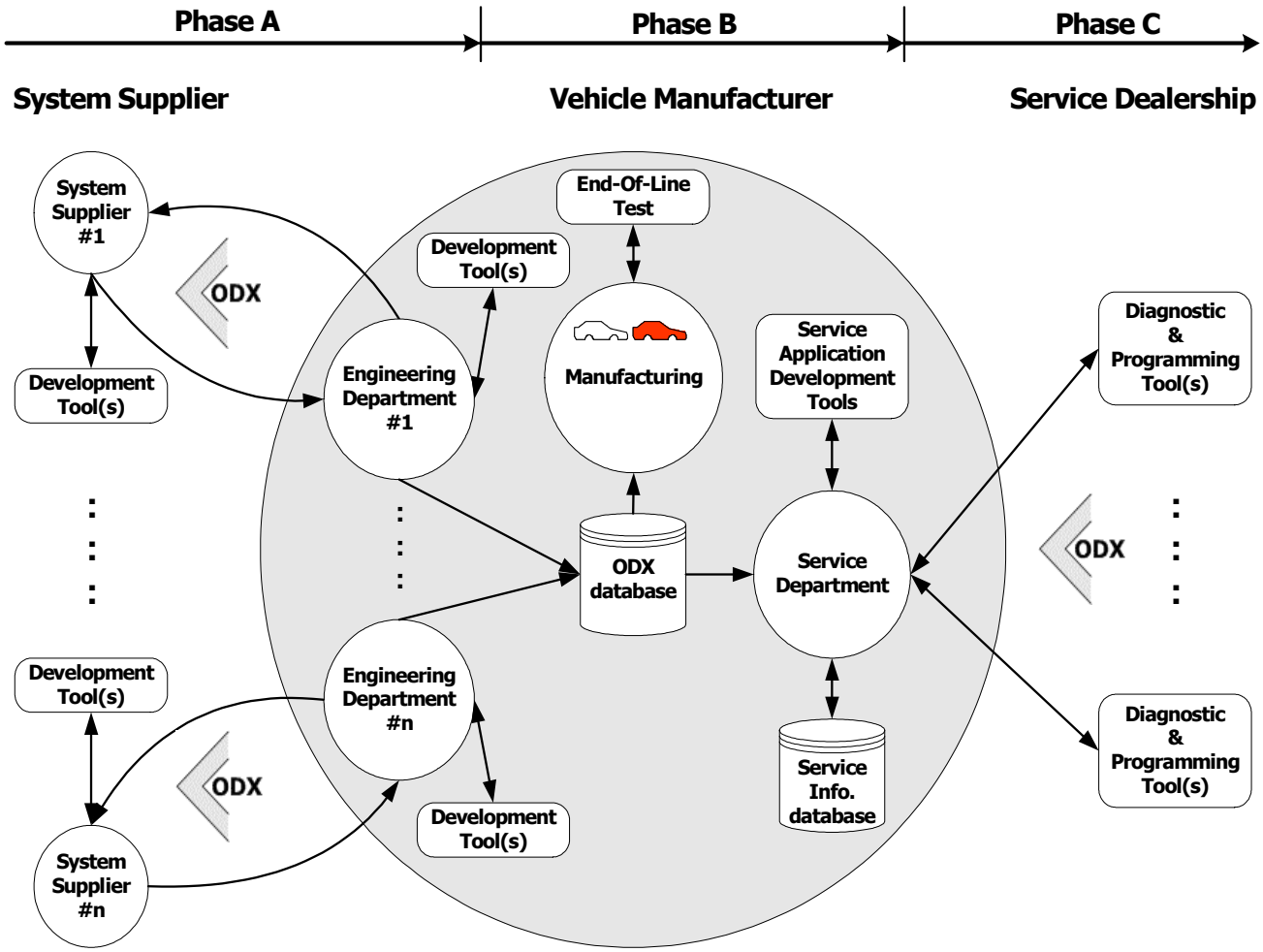


Figure 2 — Example of ODX process chain

4.2 USE CASE 2: CROSS VEHICLE PLATFORM ECU DIAGNOSTIC DEVELOPMENT

A vehicle manufacturer implements electronic systems into multiple new vehicle platforms. There is little variation in the electronic system across the different vehicle platforms. Utilising the same ECU in many different vehicle platforms reduces redundant development effort. The majority of design, normal operation, and diagnostic data of an electronic system can be reused in various vehicles.
Large automotive manufacturer tend to have multiple engineering development centres. Diagnostic data exchange can be based on the ODX data format to reduce the amount of re-authoring of diagnostic data at different development sites.
Figure 3 shows an example of cross vehicle platform ECU diagnostic development between two (2) engineering sites.

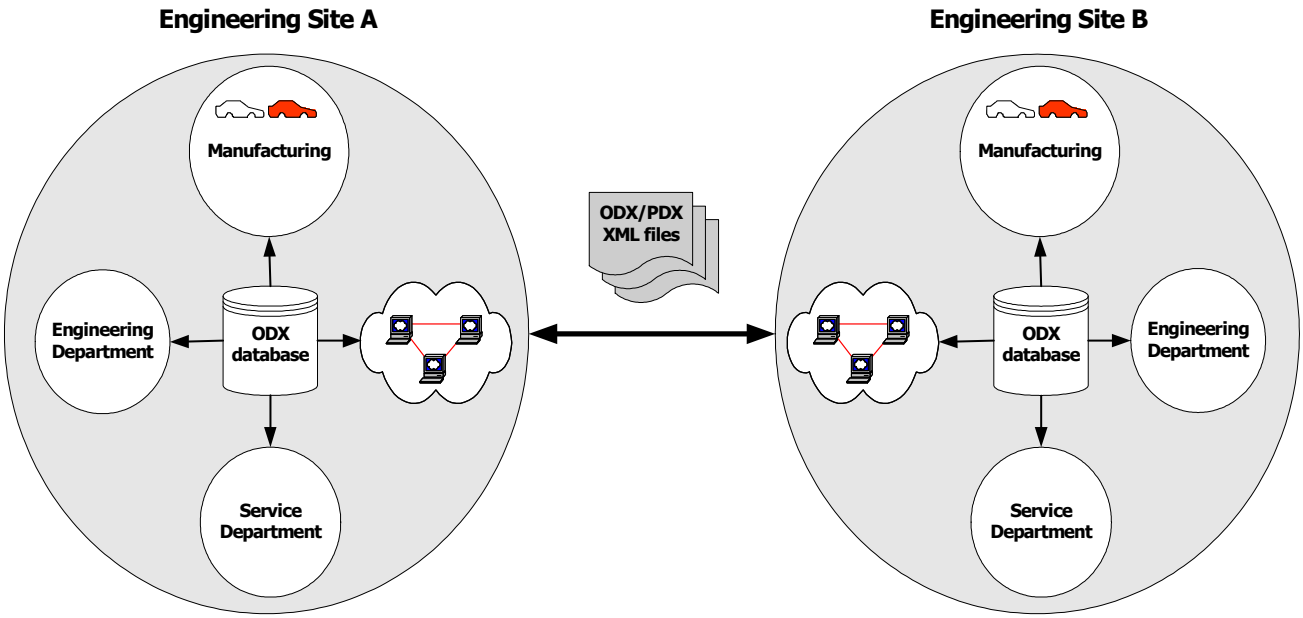


Figure 3 — Example of cross vehicle platform ECU diagnostic development

IMPORTANT — Providing diagnostic data in the ODX XML industry standard format will avoid re-authoring into various test tool specific formats at different engineering sites.

4.3 USE CASE 3: FRANCHISED AND AFTERMARKET SERVICE DEALERSHIP DIAGNOSTIC TOOL SUPPORT

Figure 4 shows one of many scenarios a vehicle manufacturer may implement to support the service dealership, franchised and aftermarket. Upon request from a dealership, which may be realized via Intranet and/or Internet the requested ODX XML files are converted into an ODX runtime format and downloaded to the dealership computer environment.

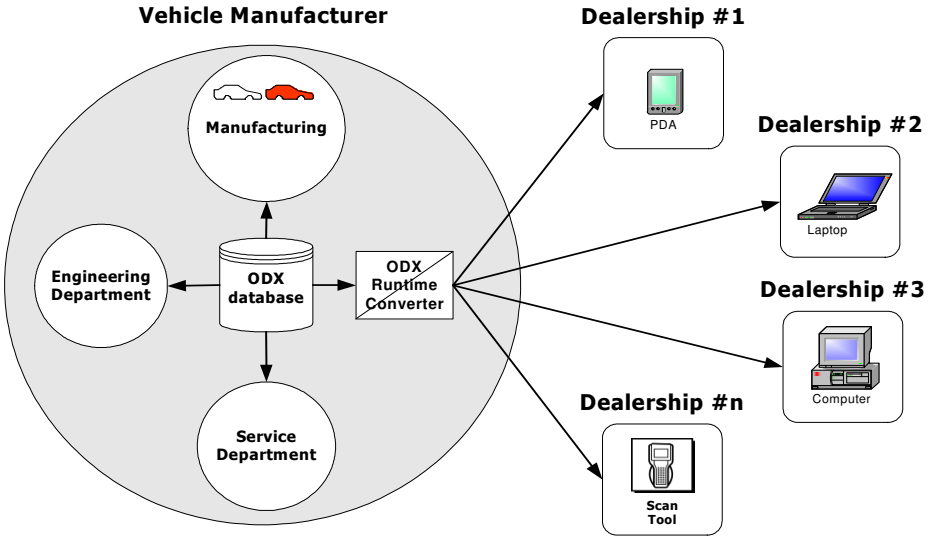


Figure 4 — Example of automotive dealership diagnostic tool support

IMPORTANT — The ODX runtime data format may be different between many diagnostic and programming applications and tools. In such case an ODX runtime converter may be used to support the data conversion to dealership specific test equipment.

4.4 ARCHITECTURE OF A RUNTIME SYSTEM

The figure below shows the interfaces of a MVCI Diagnostic Server and the location of ODX. The interface names in the figures are used throughout this document.

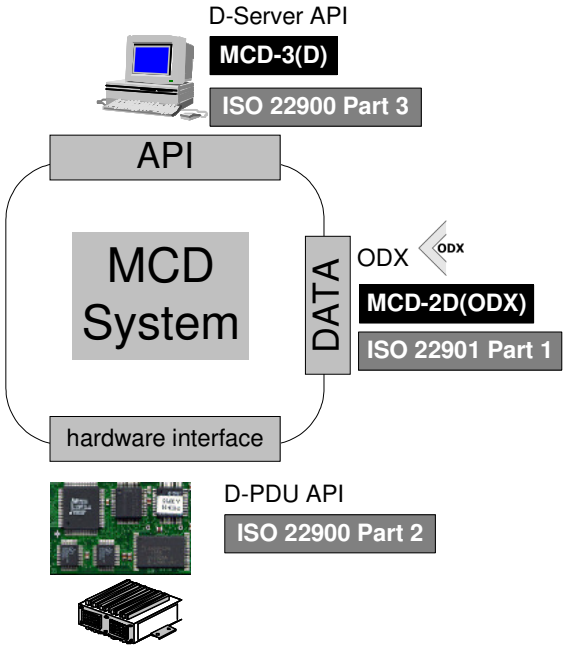


Figure 5 — Architecture of a Runtime System

4.5 ODX BENEFIT EXAMPLES

4.5.1 ECU SYSTEM SUPPLIER

The following benefits are applicable to the ECU system supplier:

- Automatic configuration of ECU diagnostic data stream & protocol,
- Documentation is generated from XML data format (ECU diagnostic content = documentation),
- Automatic configuration of development tester to verify ECU diagnostic behaviour,
- XML data format provides machine readable information to import into supplier data base,
- Generation of *.c and *.h files to configure diagnostic kernel configuration.

4.5.2 VEHICLE MANUFACTURER ENGINEERING

The following benefits are applicable to the vehicle manufacturer engineering:

- Specification and exchange of diagnostic data in standardized format,
- Reduction of diagnostic data stream authoring,
- Various development testers are supported with "single source“ data.

4.5.3 VEHICLE MANUFACTURER MANUFACTURING

The following benefits are applicable to the vehicle manufacturer manufacturing:

- Reuse of verified diagnostic data,
- End-Of-Line tester uses the same diagnostic data stream configuration data as engineering diagnostic tester.

4.5.4 VEHICLE MANUFACTURER SERVICE DEPARTMENT AND DEALERSHIPS

The following benefits are applicable to the vehicle manufacturer service department and dealerships:

- Reuse of verified diagnostic data,
- Less cost to vehicle manufacturer to distribute diagnostic data stream information,
- "Pull“ (via Intranet/Internet) diagnostic data versus "Push“ (e.g. send CD ROMs).

4.5.5 TEST EQUIPMENT MANUFACTURER

The following benefits are applicable to the test equipment manufacturer:

- High quality scan tool software,
- Focus on "killer diagnostic application(s)“ versus bits & bytes,
- Reuse of vehicle manufacturer verified diagnostic data stream information.

4.5.6 FRANCHISED AND AFTERMARKET DEALERSHIPS

The following benefits are applicable to the franchised and aftermarket dealerships:

- Reuse of OEM verified data,
- On-time availability of diagnostic data stream information,
- Simple method to download configuration data to tester,
- Download on demand versus buying tester software update upfront.

4.5.7 LEGAL AUTHORITIES

The following benefits are applicable to the legal authorities:

- Standardized description format for enhanced data stream documentation (e.g. DTCs, PIDs, etc.),
- No need for enhanced diagnostic data stream standardization,
- Fulfils requirement of making enhanced diagnostic data stream information available to aftermarket.

5 SPECIFICATION RELEASE VERSION INFORMATION

5.1 SPECIFICATION RELEASE VERSION LOCATION

The release version of the ODX standard can be obtained from every ODX file instance. It is contained in the MODEL-VERSION element.

```
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=" odx.xsd" MODEL-VERSION="2.1.0">
```

5.2 SPECIFICATION RELEASE VERSION

The specification release version of this document is: 2.1.0

6 APPLICATION OF THE UNIFIED MODELLING LANGUAGE (UML)

6.1 GENERAL ASPECTS

The Unified Modelling Language (UML) has been recognized as the leading standard for object-oriented systems design and data modelling. It is used within this standard to define the ODX data model formally and unambiguously. It also enhances the ease of understanding through graphical data model diagrams.
Every reader of this standard may not know the UML. Therefore, a short introduction is given within this chapter. This section is not a complete description of the UML. Rather, those aspects of the UML are described that are actually used within the ODX data model. Specifically, from the large set of available UML diagrams, we only apply class diagrams for data modelling within this standard.

6.2 CLASS DIAGRAMS

6.2.1 CLASS

The central modelling element within the UML is a class. A class represents a set of similar real-world objects. Generally, a class can be instantiated many times. Every instance of a class is called an object. A class has attributes defining the properties of these objects and methods defining the actions an object can perform or that can be performed on the object through another object. For this standard, methods are irrelevant and are omitted from the description.

PERSON
NAME[1] : String AGE[1] : int PROFESSION[0..1] : String

Figure 6 — UML class

Figure 6 shows the representation of a class and its attributes within the UML notation. A class is symbolized through a rectangle having one to three compartments. The top compartment contains the name of the class, e.g. PERSON. The second compartment contains the attributes of the class, e.g. NAME, AGE, PROFESSION. Methods are defined in an optional third compartment. The attribute- and method-compartments do not have to be displayed. As the method-compartment is always empty within the ODX data model diagrams, it is never displayed.
Attributes consist of the attribute name, e.g. NAME, followed by the attribute's cardinality, e.g. [1] or [0..1] and the attribute type, e.g. String. Furthermore, a default value for the attribute may be specified. Such a default value follows the type descriptor of the attribute and is separated from it by the symbol "=".
Throughout the ODX data model, class names and attribute names are written in captal letters.

6.2.2 INHERITANCE RELATIONSHIPS

Classes can inherit attributes from other classes. In Figure 7, a new class SCHOOLKID is derived from the class PERSON. This means that implicitly the class SCHOOLKID has all the same attributes as PERSON plus those that are defined specifically for the new class, e.g. GRADE. PERSON is called the parent or super class; SCHOOLKID is called the child or subclass of the inheritance relationship. Because the subclass adds more detail to the super class and is thus more specific, inheritance relationships are often called "specialisations".
Inheritance relationships can be used to build inheritance trees of arbitrary depth. A class within such a tree inherits all properties from those classes within the transitive closure of ancestors (parents, grandparents, etc.) in the inheritance tree.

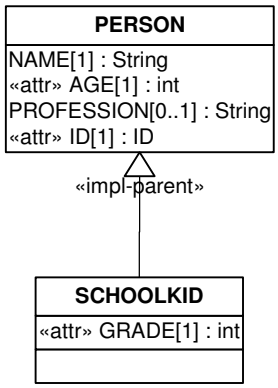


Figure 7 — Inheritance Relationship

6.2.3 INTERFACES

Interfaces are a special means to group certain aspects of a class and make other classes dependent only one of these aspect groups, instead of the class as a whole.
An interface symbol looks identical to a class symbol. It can be distinguished from class through the stereotype <<interface>> within the symbol's name compartment.

Figure 8 contains a small example of how interfaces are used within a model. A class COMPANY implements two interfaces: EMPLOYER and COMPETITOR. These are two roles of a company as observed by different tester classes (i.e. an employee and a competing other company). A person being employed by the company will not observe the company as a competitor. Equally, a competing company will not observe the competitor as an employer. If a class implements an interface, a dashed arrow shows this, similar to the inheritance arrow. A tester class of an interface (e.g. PERSON) may simply use associations or aggregations to relate to an interface. It is important to know that the tester class (PERSON) may only use those aspects of a class implementing the EMPLOYER interface that are defined within this interface. Therefore, a company as may equally employ a person by an administrative body, because the latter also implements the interface EMPLOYER.

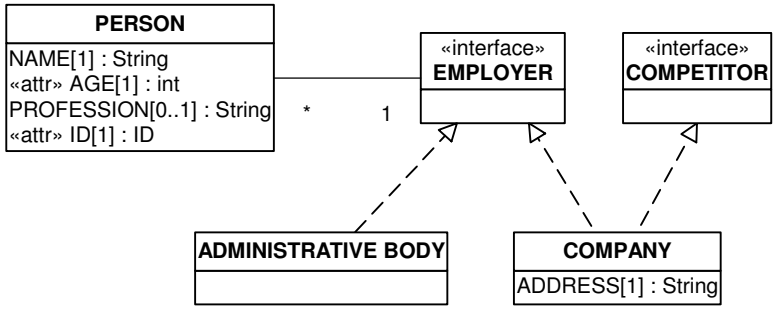


Figure 8 — Interfaces

6.2.4 AGGREGATION AND COMPOSITION RELATIONSHIPS

Besides the inheritance relationship, a pair of classes may also have an aggregation or a composition relationship between them. Aggregation or composition relationships are used, when an instance of one class is contained within an instance of another class. They are drawn as a line with a diamond at the end of the containing class. A composition relationship's diamond is filled; an aggregation's diamond is not.
The difference between these two kinds of so-called whole part relationships is as follows: The contained element (the part) within a composition relationship is fully dependent on the containing element (the whole). If the whole is deleted, the part may no longer exist. Therefore, a part may only belong to one whole, which means it only has one incoming composition relationship. An aggregation relationship is one of shared parts. This means, multiple wholes may own the same part. By consequence, the part's life cycle is not dependent on the whole's life cycle[1].

Figure 9 gives an example of three classes with composition and aggregation relationships defined between them. A PERSON may have two feet (two instances of type FOOT). A foot is generally very dependent on its owner. Therefore, a composition relationship is used. If the PERSON no longer exists, the feet will neither. In contrast, a PERSON may also have a lot of COATs. However, a COAT may generally be used by

[1] Please note that in the special case, where the data model is implemented in XML (see below), aggregation and composition relationships are both mapped onto a sub element relationship between the two model elements. However, the UML semantics guide prohibits multiple classes to have a composition relationship to the same class. Therefore, aggregation relationships are used instead, even though the implementation in XML does not differ.

multiple people and it's life-cycle is generally independent of the life-cycle of one of its wearers. By consequence, the relationship between a PERSON and a COAT may be modelled as an aggregation relationship.
Composition relationships may carry cardinalities as shown within the figure. The following cardinalities are common within UML models:

1	Exactly one (single obligate)
0..1	Zero or one (single optional)
1..*	One or many (multiple obligate)
0..* or *	Zero or many (multiple optional)

Besides, any more specific cardinality may be specified, e.g. 5 or 2..8.
Cardinalities are read as follows: To know how many instances of class PERSON are related to one instance of class FOOT, the cardinality at the PERSON end of the relationship is relevant. Vice versa: To know how many instances of class FOOT may be related to one instance of class PERSON, the cardinality at the FOOT end of the relationship is relevant. This yields the following result: One instance of class FOOT may be related to exactly one instance of class PERSON (A foot always belong to one and only one person) and one instance of class PERSON must be related to exactly two instances of class FOOT (a person usually has two feet).

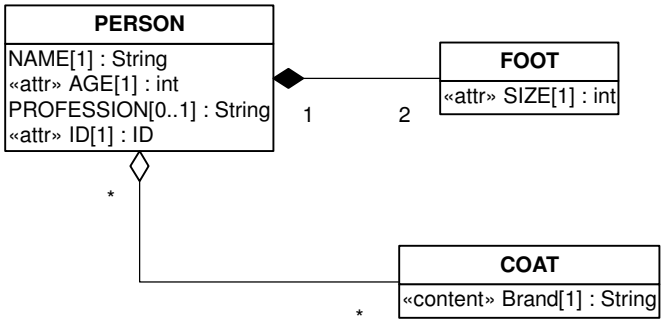


Figure 9 — Composition and Aggregation Relationships

6.2.5 ASSOCIATION RELATIONSHIPS

The third kind of class relationship used within the ODX data model is the association relationship. It is drawn as a simple line between two classes. An association relationship has weaker semantics than the composition relationship. Here, both classes have "equal rights". An association relationship opens visibility onto the properties (attributes, methods) of the related class. With composition relationships, an instance of a composed class is only contained within one instance of the composing class. In contrast, instances of associated classes may be related to many instances of other classes. To restrict the number of instances being related to each other, cardinalities are used in the same manner as with composition relationships.

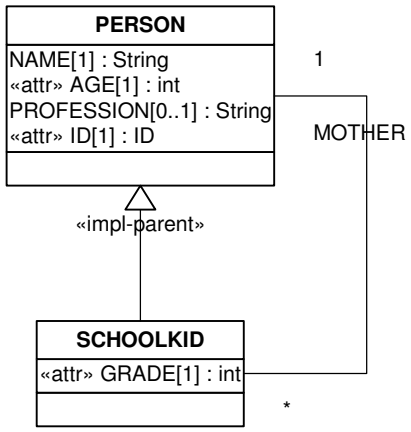


Figure 10 — Association Relationship

Associations may carry a name and a role name can be given at every association end. In Figure 10, the PERSON class carries the role MOTHER in a relationship to a SCHOOLKID. The cardinalities state that a SCHOOLKID has exactly one mother, whereas a PERSON may be the MOTHER or an arbitrary number of SCHOOLKIDs.
An association relationship may carry an arrow at one association end, which represents a navigation direction, if applicable. This allows defining instances of which of the two classes may actually access instances of the other class.

6.2.6 ASSOCIATION CLASSES

Association classes are a hybrid of an association and a class. Generally, they can be interpreted as an association carrying its own attributes (and methods). They are drawn like an association with a class symbol being connected to the centre of the association with a dashed line.
The example of a PERSON being employed by an EMPLOYER is modelled as an association class in Figure 11. Here, the employment relationship is attributed with SALARY. This employment-specific salary is clearly not a property of a PERSON, because one person may have multiple salaries, if it is multiply employed. In addition, the salary may suddenly not exist, if the PERSON is unemployed. It is also not a property of the EMPLOYER, because the EMPLOYER has many different employee-specific salaries. This is the best indication that the SALARY is a property of an employment relationship being established between a PERSON and an EMPLOYER.

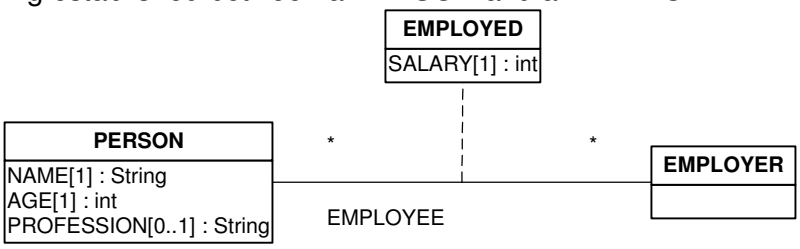


Figure 11 — Association Class

6.2.7 CONSTRAINTS

The UML allows defining constraints on the model elements used. Generally, constraints can be freely defined. However, the ODX standard only uses these features scarcely. One constraint that is used frequently is the {xor} constraint between two associations (or aggregations). It enforces that an instance of one class has a relationship with instances of one or the other associated class but never both at the same time.
Figure 12 displays an example of such a constraint. A person wears either two SHOEs or two BOOTs at any one time, but never both together.

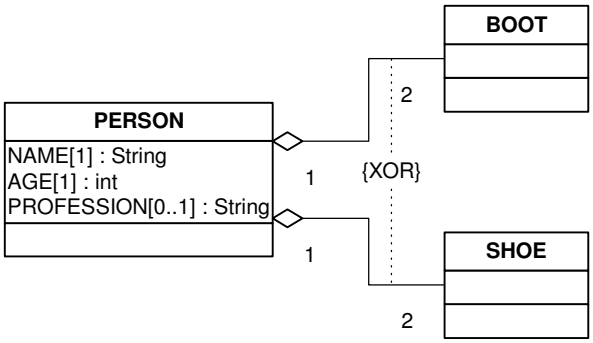


Figure 12 — Constraint

6.3 MODEL STRUCTURING WITH PACKAGES

To structure the ODX data model, UML packages are used. Packages define a separate namespace and may contain any other kind of UML model element or diagram. Packages are used to organize the model into distinguishable units.
Between packages, dependencies of type <<import>> are used. Such a dependency makes model elements (e.g. classes) contained within one package visible to another package.
Figure 13 shows an example of three packages: PERSON, COMPANY and EMPLOYMENT. With respect to our example, the classes PERSON and SCHOOLKID together with the MOTHER relationship would be defined in package PERSON. COMPANY and ADMINISTRATIVE-BODY could be defined in package COMPANY, while the interface EMPLOYER, and the association class EMPLOYED would be part of the EMPLOYMENT package. As this latter packages needs access to some of the classes defined in the previous packages, an <<import>> dependency to these packages is modelled.
The structure of the complete ODX model is shown within package diagrams. The contents of the packages are not shown within these diagrams. Rather, separate class diagrams are used to show the contents of one package. Occasionally, one package contains multiple class diagrams to show various aspects of the contained model and keep the class diagrams simple.

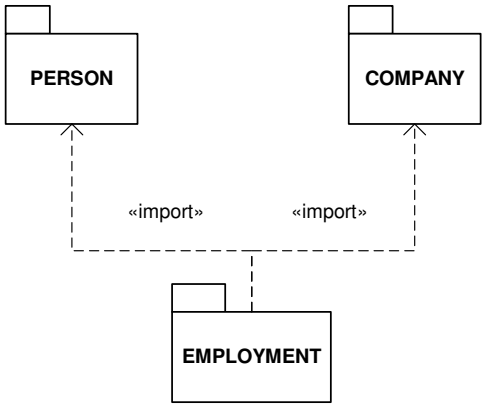


Figure 13 — Packages

6.4 MAPPING TO XML

The target implementation format of the ODX exchange format is XML. In this paragraph, a very short idea of how the UML model is mapped onto XML language elements is given. It does not claim do be complete, but is only an example that simplifies the comprehension of the XML examples given throughout this document. The complete XML mapping rules and the resulting XML Schema are described in Chapter 8.
Generally, the following rules apply:

d) A class is generally mapped onto an XML element.

e) Attributes of a class are mapped onto XML attributes of the XML element, if the <<attr>> stereotype is defined for the UML attribute or onto a XML sub element, if no stereotype is defined. If an UML attribute with stereotype <<content>> is defined, the attribute is simply mapped onto the content of the XML element.

f) Classes connected to another class via a composition or aggregation relationship are mapped onto sub elements. In cases, where the target cardinality of the contained class is multiple, a wrapper element around the sub elements is generated. The wrapper element is omitted, if the {nowrapper}-constraint is defined on the composition or aggregation relationship respectively.

g) Associations to other classes are mapped onto XML references to other elements. Two kinds of reference mechanisms have been defined for ODX. These are marked through stereotypes <<snref>> and <<odxlink>>, respectively. For the detailed implementation of these reference mechanisms, please refer to Section 7.3.13.

h) In ODX, classes that are part of an inheritance hierarchy are mapped in XML depending on which UML inheritance relationship stereotype is used. The two inheritance stereotypes used in ODX are <<impl-child>> and <<impl-parent>>. Both stereotypes represent specialisation as defined in 5.2.2. When specifying an inheritance relationship using the <<impl-child>> stereotype, all child or sub-types are implemented in XML; i.e., every child type will have one XML Schema type. When using the <<impl-parent>> stereotype, the generated XML element carries an attribute of name xsi:type. It contains the name of the child-type that is used within the particular instance ,.

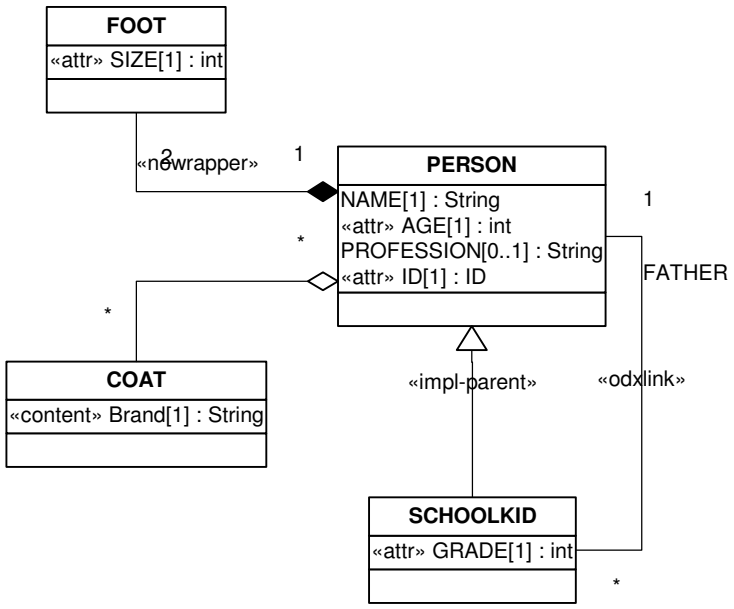


Figure 14 — Mapping Example

These rules cover aspects of the mapping in an incomplete manner. However, they suffice to comprehend the examples in the following chapters. According to these rules, the example in Figure 14 is mapped onto the XML document shown below. The <EXAMPLE> root element has been included to satisfy well-formed ness regulations of the XML.

```
<EXAMPLE>
	<PERSON ID="ID_64711" AGE="33" xsi:type="PERSON">
		<NAME> Jack O. Trades </NAME>
		<PROFESSION> All Duty Service Technician </PROFESSION>
		<COAT> Gucci </COAT>
		<FOOT SIZE="8"></FOOT>
		<FOOT SIZE ="9"></FOOT>
	</PERSON>
	<PERSON ID="ID_60815" AGE = "8" xsi:type="SCHOOLKID" GRADE="3">
		<NAME> Kevin </NAME>
		<FOOT SIZE="5"></FOOT>
		<FOOT SIZE="5"></FOOT>
		<FATHER-REF ID-REF="ID_64711"/>
	</PERSON>
</EXAMPLE>
```

Throughout the document some other constraints and stereotypes are used that have not been explained within this section. These include the following:

<<value-inherit>> for inheritance between diagnostic layers;

<<element-name>> to deviate from the standard naming schema that XML elements are named like their UML class counterparts;

<<no-wrapper>> to specify that the standard mapping of generating a XML wrapper element around multiple XML sub elements of the same kind is not applied;

<<import>> to describe the import of data from another diagnostic layer;

{ordered} to enforce that the order of XML sub elements is significant and must not be changed by any ODX-compliant tool;

If more detailed information on the mapping to XML is needed, please refer to chapter 8.

7 ODX DATA MODEL

7.1 GENERAL MODELLING PRINCIPLES

7.1.1 COMMON MEMBERS

SHORT-NAME identifies an ODX object. Its length is limited to 128 characters. A short name consists of letters, digits and "_" character. The following regular expression describes the syntax of short names:
[a-zA-Z0-9_]+
LONG-NAME is a short description of the functionality of the ODX object. Its length is limited to 255 characters. The LONG-NAME should be displayed in human interface of an application instead of SHORT-NAME.
DESC describes detailed the functionality of the ODX object and has no length restriction. This element is optional and may involve several paragraphs marked by the element <p>. Following tags known from HTML can be used to format the text:
, <i>, <b>, <u>, <sub>, <sup>, <ul>, <ol>, <li>.
LONG-NAME and DESC may additionally have an optional member TI (text identifier) that supports multilingualism for different kinds of text module. The mapping technology is tool-/manufacturer-specific. For example, they may occur in an external file. The TI attribute allows then the mapping between external and internal descriptions and can be used for language translations.
ELEMENT-ID is used as a common type to represent the above listed members throughout the ODX data model. An attribute HANDLE of this type is used at all elements using these common members.
ID is an identifier used by the linking concept "odx-link" described later in this document. The value of ID is restricted by XML specification. Any ODX compliant tool shall not change the content of the ID over the whole life cycle of the data element. Any system that provides import/export facilities of ODX into an internal format shall ensure to maintain the IDs during an import / modify / export cycle. The process owner is responsible to ensure the uniqueness of the ID throughout the overall process chain. That means the tool can of course assign a new ID to a new object. The method being used to ensure uniqueness of IDs in order to avoid the necessity of subsequent changes due to conflicts is up to the process owner.
However the usage of Universally Unique Identifiers (UUIDs) as described in ISO/IEC 11578:1996 is recommended.
OID is used for invariant identification of an object but not for linking. Any ODX compliant tool shall not change the content of the Object-ID (if present) over the whole life cycle of the data element. Any system that provides import/export facilities of ODX into an internal format shall ensure to maintain the OIDs during an import / modify / export cycle. The process owner is responsible to ensure the uniqueness of the OID throughout the overall process chain. The usage of Universally Unique Identifiers (UUIDs) as described in ISO/IEC 11578:1996 is recommended.

7.1.2 COMMON OBJECTS

7.1.2.1 SPECIAL DATA GROUP

Special data groups (SDGs) are the standard extension mechanism of ODX; they are used to store any kind of data that is not covered by the standardized part of the data model in a structured way. Examples for the usage of SDGs in ODX are the company-specific documentation in COMPANY-DOC-INFO. The ODX specification defines the structure of SDGs but not their content; thus, a standard diagnostic tool conforming to ODX is not required to process SDGs.

Drawing: SpecialData

Package: AdminInformation::AdminData

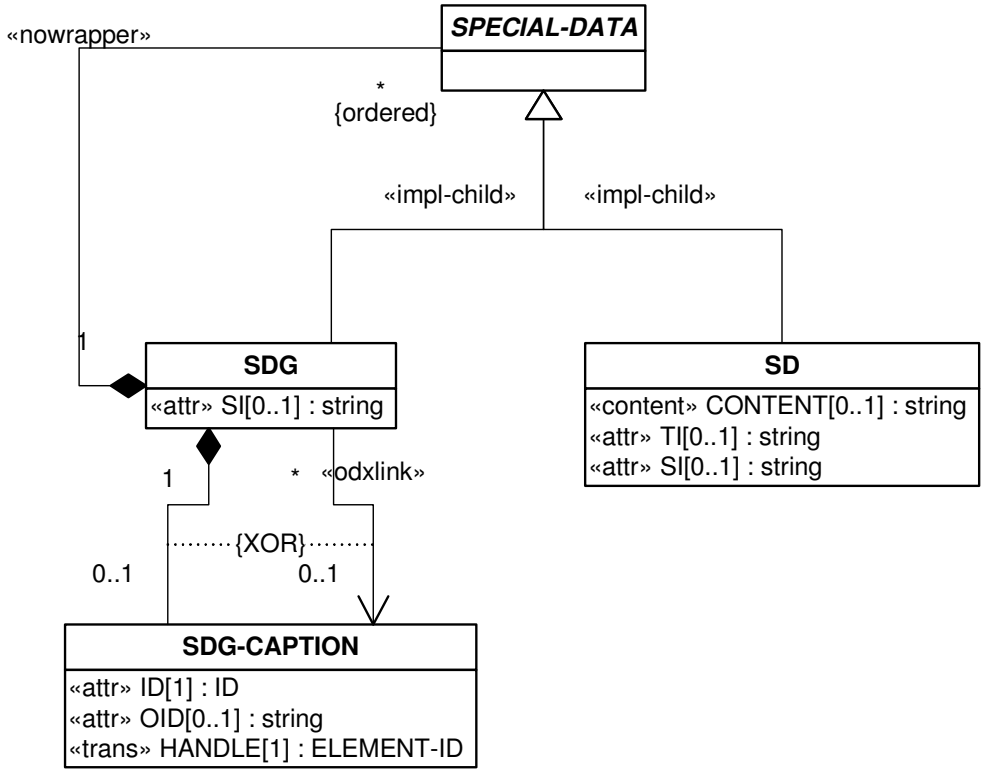


Figure 15 — Special Data Group (SDG)

An SDG contains an optional CAPTION to describe the SDG content, and a list of SDG and SD objects that contain the special data. This list can contain an arbitrary number of SDGs and SDs, and the ordering of these SDGs and SDs is not restricted in any way. SDGs can be nested recursively; that way, very complex data structures may be defined as SDGs. The member SI is used to add semantic information to the appropriate object. It can be used for example to implement a table with key-value pairs (see example below). The reuse of an already defined SDG-CAPTION can be done with the SDG-CAPTION-REF, which results from the odxlink between SDG and SDG-CAPTION.

EXAMPLE Definition of tables of property lists with SDGs

```
<SDGS>
      <SDG>
            <SDG-CAPTION ID="id123">
```

```
                    <SHORT-NAME>properties</SHORT-NAME>
                    <LONG-NAME>A table of key-value pairs</LONG-NAME>
             </SDG-CAPTION>
             <SD SI="part-number">4711</SD>
             <SD SI="index">007</SD>
             <SD SI="SAP-number">1234567</SD>
      </SDG>
</SDGS>
```

EXAMPLE Describing deep substructures with SDGs (fictitious example)

With SGDs it is possible to create complex structures of any depth. A number of tables, for example, can be combined to build a 3-dimensional matrix:

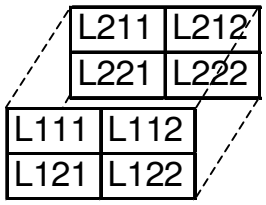


Figure 16 — Special Data Group

This matrix can be described by an SDG structure of depth 3:

```
<SDGS SI="example">
      <SDG>
             <SDG-CAPTION ID="imatrix">
                    <SHORT-NAME>matrix</SHORT-NAME>
                    <LONG-NAME>A 3-dimensional matrix mapped to an SDG
structure</LONG-NAME>
             </SDG-CAPTION>
             <SDG>
                    <SDG-CAPTION ID="itable1">
                           <SHORT-NAME>firstTable</SHORT-NAME>
                    </SDG-CAPTION>
                    <SDG>
                           <SDG-CAPTION ID="i1row1">
                                  <SHORT-NAME>firstRow</SHORT-NAME>
                           </SDG-CAPTION>
                           <SD SI="example">L111</SD>
                           <SD SI="example">L112</SD>
                    </SDG>
                    <SDG>
                           <SDG-CAPTION ID="i1row2">
                                  <SHORT-NAME>secondRow</SHORT-NAME>
                           </SDG-CAPTION>
                           <SD SI="example">L121</SD>
                           <SD SI="example">L122</SD>
                    </SDG>
             </SDG>
             <SDG>
                    <SDG-CAPTION ID="itable2">
                           <SHORT-NAME>secondTable</SHORT-NAME>
                    </SDG-CAPTION>
                    <SDG>
                           <SDG-CAPTION ID="i2row1">
                                  <SHORT-NAME>firstRow</SHORT-NAME>
                           </SDG-CAPTION>
                           <SD SI="example">L211</SD>
                           <SD SI="example">L212</SD>
                    </SDG>
                    <SDG>
```

```
                        <SDG-CAPTION ID="i2row2">
                              <SHORT-NAME>secondRow</SHORT-NAME>
                        </SDG-CAPTION>
                        <SD SI="example">L221</SD>
                        <SD SI="example">L222</SD>
                    </SDG>
                </SDG>
            </SDG>
</SDGS>
```

7.1.2.2 AUDIENCE AND ADDITIONAL-AUDIENCE

Audiences define the target group or groups of users for the following elements:

- DIAG-COMM
- MULTIPLE-ECU-JOB
- CONFIG-ITEM, CONFIG-RECORD and DATA-RECORD of ECU-CONFIG
- BASE-FUNCTION-NODE of FUNCTION-DICTIONARY-SPEC
- SESSION-DESC and DATABLOCK of ECU-MEM

Five groups are predefined in ODX and thus five values can be specified there:

- IS-SUPPLIER,
- IS-DEVELOPMENT,
- IS-MANUFACTURING,
- IS-AFTERSALES and
- IS-AFTERMARKET.

Each value can be set to "true" or "false". By default the elements potentially carrying an audience are available for every group, if the audience element is empty or not present. To expand the list of enabled or disabled groups the subelement ADDITIONAL-AUDIENCE of the DIAG-LAYER can be used. ADDITIONAL-AUDIENCE is an individual list of users that shall be enabled or disabled to read the corresponding diagnostic elements. A diagnostic element may contain either enabling or disabling references to ADDITIONAL-AUDIENCEs.

Drawing: LayerCommon
Package: DiagLayerStructure

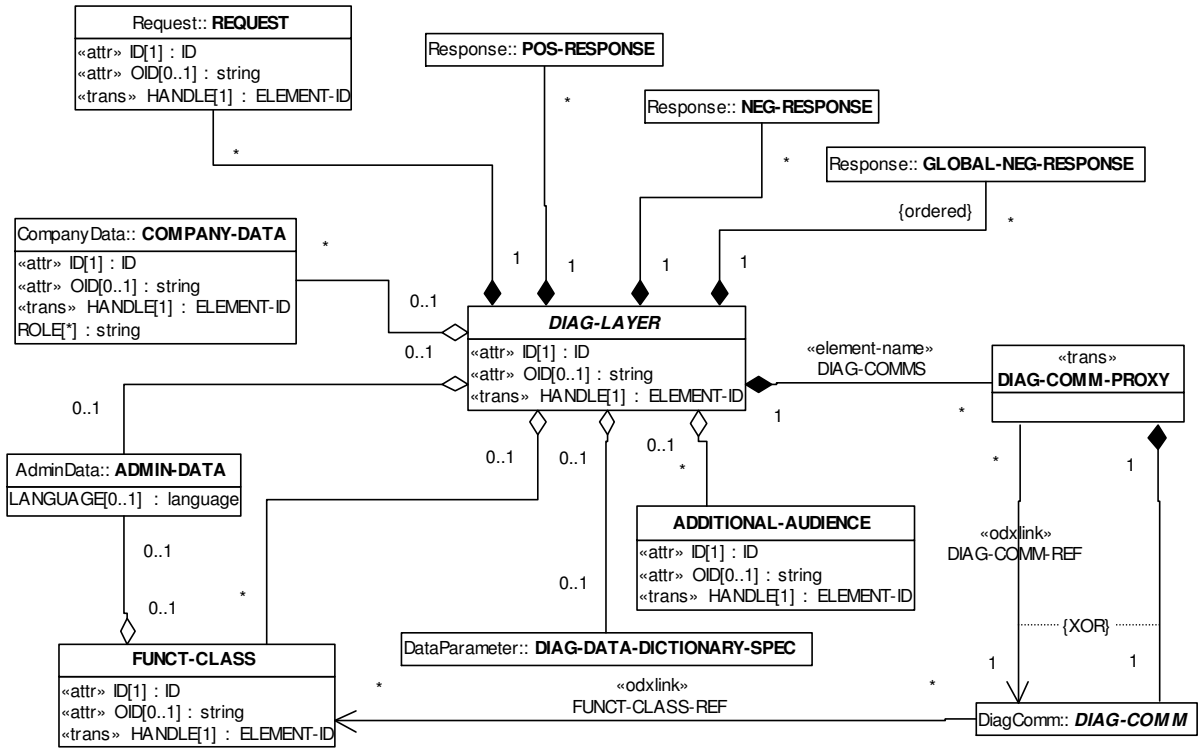


Figure 17 — Layer common – additional audience

Example of ADDITONAL-AUDIENCES:

```
< ADDITIONAL-AUDIENCES>
       < ADDITIONAL-AUDIENCE ID="IS-SUPPLIER.ID">
              <SHORT-NAME>IS-SUPPLIER</SHORT-NAME>
              <LONG-NAME>IS-SUPPLIER</LONG-NAME>
</ ADDITIONAL-AUDIENCE>
< ADDITIONAL-AUDIENCE ID="IS-DEVELOPMENT.ID">
<SHORT-NAME>IS-DEVELOPMENT</SHORT-NAME>
<LONG-NAME>IS-DEVELOPMENT</LONG-NAME>
       </ ADDITIONAL-AUDIENCE>
       < ADDITIONAL-AUDIENCE ID="MANUFACTURER_C.ID">
       <SHORT-NAME>MANUFACTURER_C</SHORT-NAME>
              <LONG-NAME>Manufacturer_C</LONG-NAME>
        </ ADDITIONAL-AUDIENCE>
       < ADDITIONAL-AUDIENCE ID="AUDI.ID">
              <SHORT-NAME>AUDI</SHORT-NAME>
              <LONG-NAME>AUDI</LONG-NAME>
       </ ADDITIONAL-AUDIENCE>
</ ADDITIONAL-AUDIENCES>
```

The “ENABLED-AUDIENCE-REF” has the meaning that this element e.g. DIAG-COMM is only available for the referenced ADDITIONAL-AUDIENCEs .
THE “DISABLED-AUDIENCE-REF” has the semantics that this element is not available for the referenced ADDITIONAL-AUDIENCEs but available for all other listed ADDITIONAL-AUDIENCEs.
Additionally to the reference ADDITIONAL-AUDIENCEs the predefined AUDIENCEs have to be evaluated by a tool.

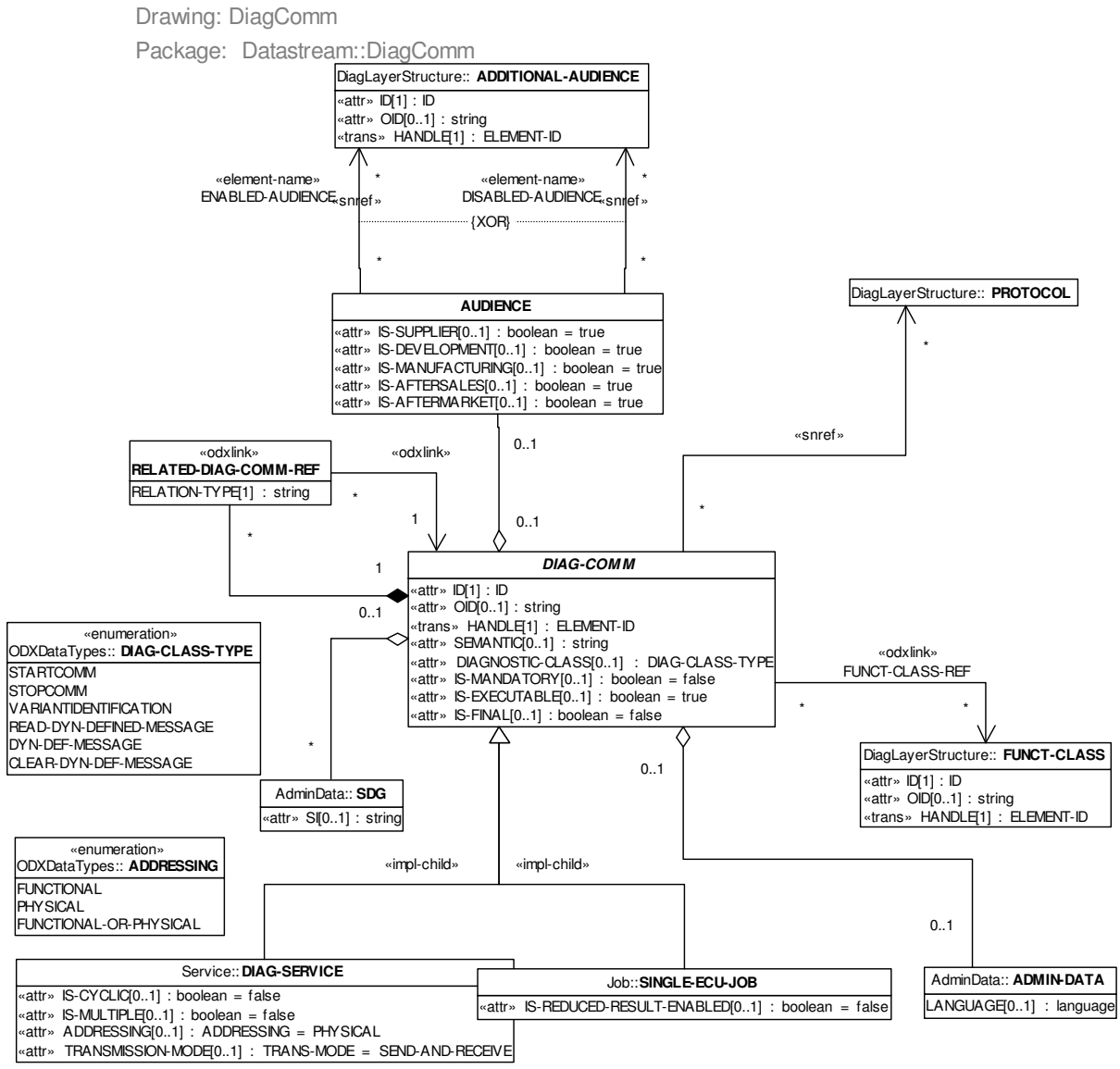


Figure 18 — DIAG-COMM and ADDITONAL-AUDIENCE

Example:

Definition of ADDITIONAL-AUDIENCES

```
<ADDITIONAL-AUDIENCES>
  <ADDITIONAL-AUDIENCE ID="MANUFACTURER_C.ID">
    <SHORT-NAME>MANUFACTURER_C</SHORT-NAME>
    <LONG-NAME>Manufacturer_C</LONG-NAME>
  </ADDITIONAL-AUDIENCE >
  <ADDITIONAL-AUDIENCE ID=„SUPPLIER_G.ID">
    <SHORT-NAME>SUPPLIER_G</SHORT-NAME>
    <LONG-NAME>SUPPLIER_G</LONG-NAME>
  </ADDITIONAL-AUDIENCE >
  <ADDITIONAL-AUDIENCE ID=„MANUFACTURER_S.ID">
    <SHORT-NAME>MANUFACTURER_S</SHORT-NAME>
```

```
    <LONG-NAME>Manufacturer_S</LONG-NAME>
  </ADDITIONAL-AUDIENCE >
  <ADDITIONAL-AUDIENCE ID="SUPPLIER_B.ID">
    <SHORT-NAME>SUPPLIER_B</SHORT-NAME>
    <LONG-NAME>SUPPLIER_B</LONG-NAME>
  </ADDITIONAL-AUDIENCE >
</ADDITIONAL-AUDIENCES >

Reference of ADDITIONAL-AUDIENCES

<DIAG-SERVICE ID="SERIVCE1.ID">
  <SHORT-NAME>SERVICE_1</SHORT-NAME>
  <AUDIENCE IS-AFTERMARKET=FALSE>
    <ENABLED-AUDIENCE-REFS>
        <ENABLED-AUDIENCE-REF SHORT-NAME=„MANUFACTURER_C"/>
    </ENABLED-AUDIENCE-REFS>
  </AUDIENCE>
</DIAG-SERVICE>
<DIAG-SERVICE ID="SERIVCE2.ID">
  <SHORT-NAME>SERVICE_2</SHORT-NAME>
  <AUDIENCE>
    <DISABLED-AUDIENCE-REFS>
        <DISABLED-AUDIENCE-REF SHORT-NAME==„SUPPLIER_B"/>
    </DISABLED-AUDIENCE-REFS>
 </AUDIENCE>
</DIAG-SERVICE>
```

The result is as follows:

- Service_1 is enabled for MANUFACTURER_C but disabled for all other defined ADDITIONAL-AUDIENCE.
- Service_2 is disabled for SUPPLIER_B but enabled for all other definded ADDITIONAL-AUDIENCE.

7.1.2.3 ADMINISTRATIVE INFORMATION

Some ODX objects like diagnostic layers or services have a very long lifetime. They may be subject to many changes possibly performed by different people from different companies. The change history of such an object should be stored with the object. Some information about the person who is responsible for the change should be available, too; that way, questions and feedback can be addressed to the right recipient. This kind of data is called "administrative data" (or short "admin data").
Each process partner can also add his own company-specific admin data to an ODX object, e.g. revision information that is needed to handle the ODX object with a content management system or a database.

Drawing: AdminData
Package: AdminInformation::AdminData

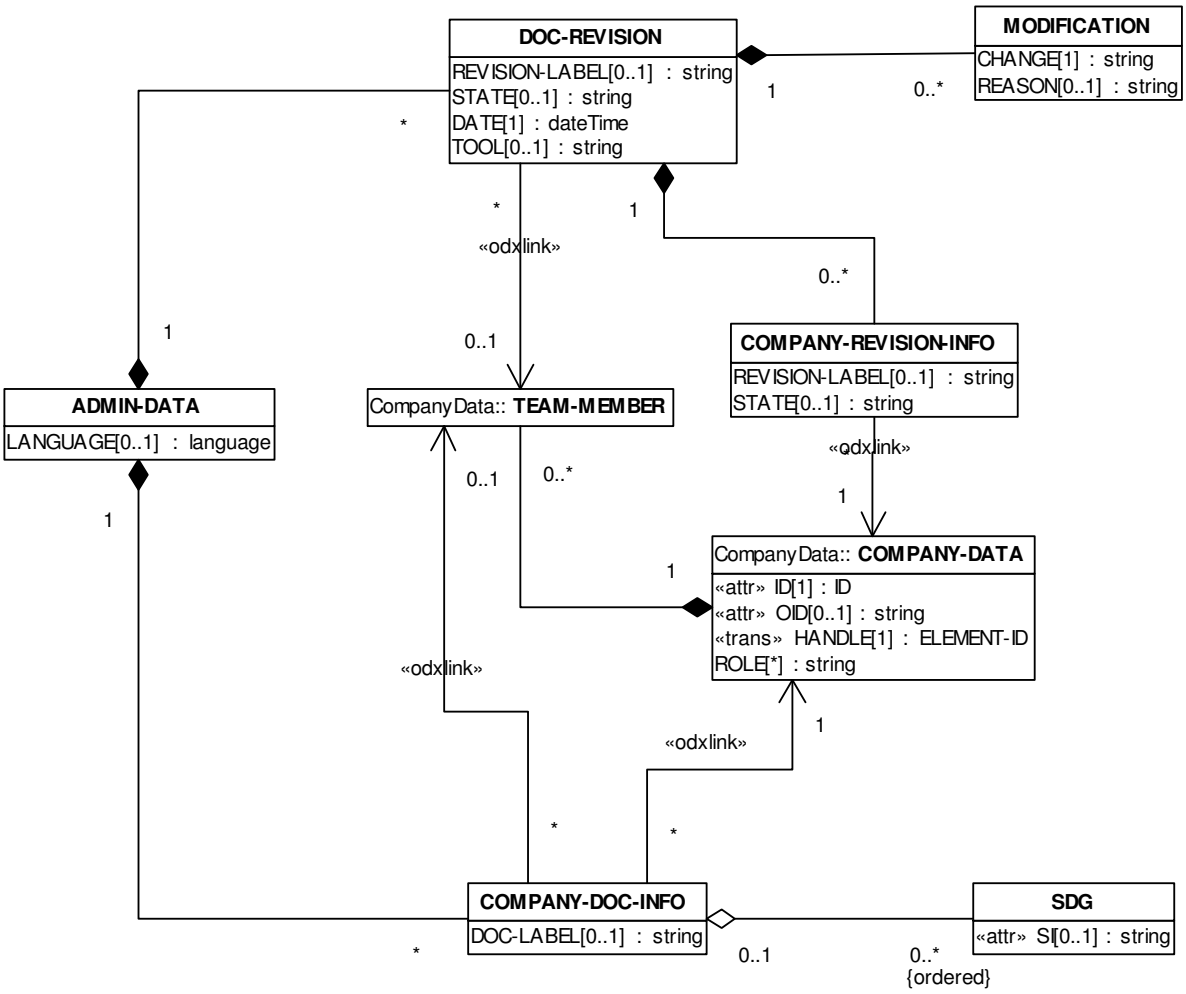


Figure 19 — Admin data

Location in ODX data model

ODX provides two classes of objects for admin data: ADMIN-DATA and COMPANY-DATA.
An ADMIN-DATA object contains the admin data that is specific to the ODX object it belongs to. Information about the companies that have created or modified the ODX object is stored in COMPANY-DATA objects. There should be a separate COMPANY-DATA object for each company that is involved as a process partner in the creation or modification of ODX objects.

Drawing: AdminInfo
Package: AdminInformation

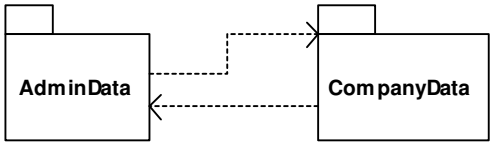


Figure 20 — Admin info

ADMIN-DATA objects are associated to a number of ODX objects. These ODX objects are typically expected to be handled (e.g. modified or exchanged) as a whole in the engineering process.
Each COMPANY-DATA object provides information about a company and its employees who participate in the engineering process. All company-specific admin data refers directly or indirectly to such a COMPANY-DATA object. With these references a process partner can filter out his own company-specific admin data.
COMPANY-DATA is placed in those ODX objects e.g. DIAG-LAYER-CONTAINER that are used as data exchange units and are possibly transferred to other process partners.
NOTE An ODX container catalogue described in chapter 8 may also contain ADMIN-DATA and COMPANY-DATA objects. Admin data from an ODX container catalogue describes revisions of the files in an ODX container; it is typically used in the data exchange process.

Structure of ADMIN-DATA

ADMIN-DATA consists mainly of two parts: DOC-REVISIONS and COMPANY-DOC-INFOS.
The natural LANGUAGE of the text data within an ADMIN-DATA object can be specified if necessary, e.g. to support cooperation with international process partners. It is a four-letter code. As an example, LANGUAGE may be set to "de_DE" for German or"de_CH" for German, swiss variant..
Each DOC-REVISION object describes one revision of the object the ADMIN-DATA belongs to. A DOC-REVISION object must contain the DATE of the revision. The person that created the revision can be referenced with a TEAM-MEMBER-REF. Optionally, the REVISION-LABEL, the STATE of the revision, the used TOOL, and the MODIFICATIONS made can be added. The process partners must agree on the usage of these objects; for example, all process partners have to apply the same revisioning scheme if they want to use the REVISION-LABEL. For each MODIFICATION the CHANGE that was made has to be described. A REASON for the change may be added.
A DATE is formatted according to the syntax rules for the DATETIME datatype in XML schema: Representation of dates and times". A syntactically correct DATE has the form CCYY-MM-DDThh:mm:ss, where CC is the century, YY the year of the century, MM the month of the year, DD the day of the month, T the separator between date and time, hh the hours of the day, mm the minutes and ss the seconds. (e.g. 2003-08-22T15:40:25).
Additionally, company-specific revision information can be stored in COMPANY-REVISION-INFO objects. This way, each process partner can attach revision information required for his individual version control process to the ODX object.
The COMPANY-DOC-INFO objects contain company-specific information about the object the ADMIN-DATA belongs to. They are mainly used for documentation purposes.
Each COMPANY-DOC-INFO object contains a reference to the COMPANY that owns it. An optional reference to the TEAM-MEMBER that is responsible for this part of the

document may be added, as well as a DOC-LABEL that can be used for identification of the COMPANY-DOC-INFO object. Arbitrarily structured content (e.g. a table that is to be included in a reference manual) can be stored in special data groups (SDGs). SDGs are described in further detail in chapter 7.1.2.1.

EXAMPLE Admin data

```
<ADMIN-DATA>
	<LANGUAGE>en_UK</LANGUAGE>
	<COMPANY-DOC-INFOS>
		<COMPANY-DOC-INFO>
			<COMPANY-DATA-REF ID-REF="SampleCompany"/>
		</COMPANY-DOC-INFO>
	</COMPANY-DOC-INFOS>
	<DOC-REVISIONS>
		<DOC-REVISION>
			<TEAM-MEMBER-REF ID-REF="JDo"/>
			<REVISION-LABEL>0.1</REVISION-LABEL>
			<STATE>INITIAL</STATE>
			<DATE>2004-02-25T15:00:00</DATE>
			<TOOL>XMLSPY</TOOL>
			<MODIFICATIONS>
				<MODIFICATION>
					<CHANGE>Initial. Addition of Example
Services</CHANGE>
					<REASON>ODX Documentation</REASON>
				</MODIFICATION>
			</MODIFICATIONS>
		</DOC-REVISION>
		<DOC-REVISION>
			<TEAM-MEMBER-REF ID-REF="JSm"/>
			<REVISION-LABEL>0.2</REVISION-LABEL>
			<STATE>REVIEW</STATE>
			<DATE>2003-11-06T15:00:00</DATE>
			<MODIFICATIONS>
				<MODIFICATION>
					<CHANGE>Review/Changes caused by Review</CHANGE>
				</MODIFICATION>
			</MODIFICATIONS>
		</DOC-REVISION>
	</DOC-REVISIONS>
</ADMIN-DATA>
```

Drawing: CompanyData

Package: AdminInformation::CompanyData

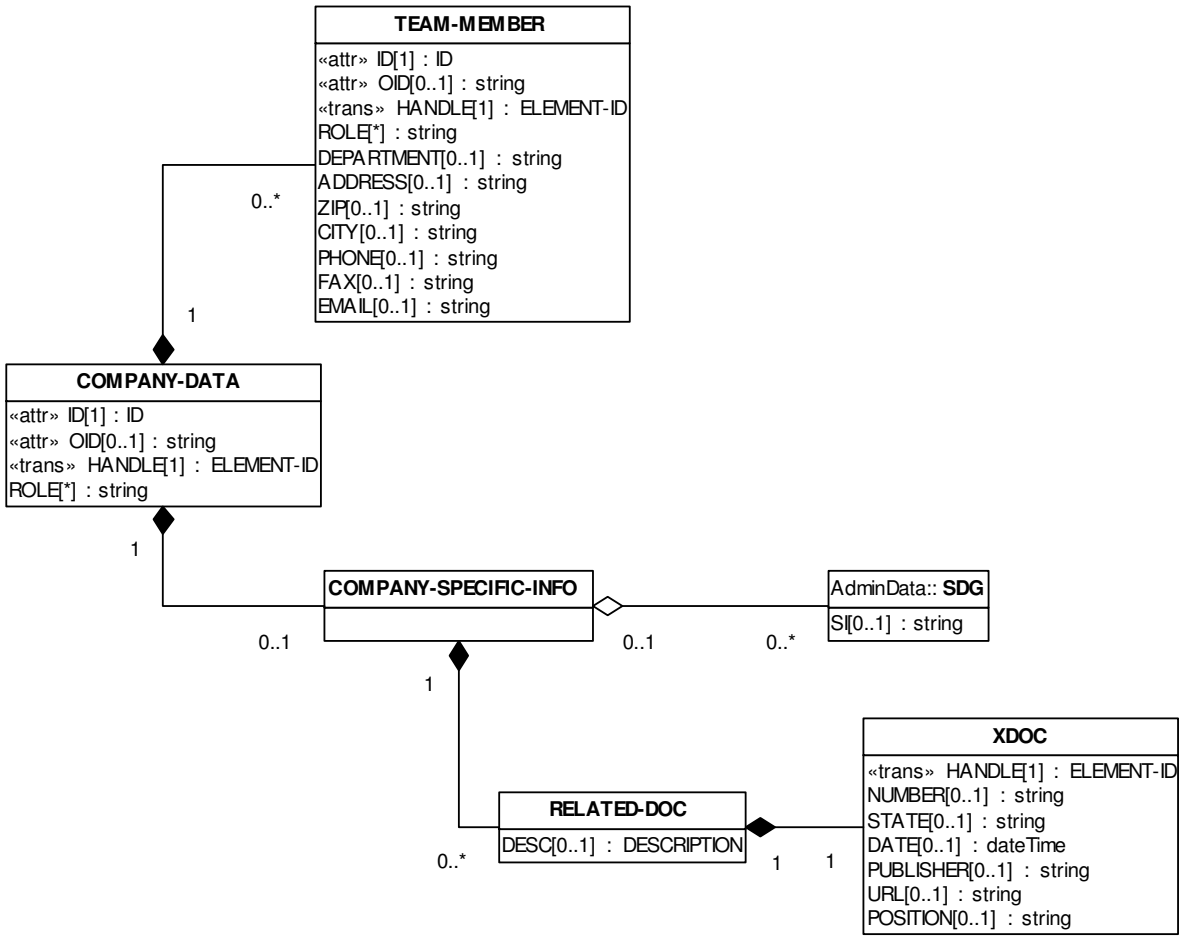


Figure 21 — Company data

Structure of COMPANY-DATA

For each employee a TEAM-MEMBER object is stored. Detailed contact information may be added with optional DEPARTMENT, ADDRESS, ZIP code, CITY, PHONE, FAX, EMAIL and ROLE objects.
Information that is not associated with a single person may be stored in the COMPANY-SPECIFIC-INFO object. RELATED-DOC objects are used for cross-referencing additional documents. For this purpose, an XDOC object is specified that may contain the optional objects NUMBER, STATE, DATE, PUBLISHER, URL, and POSITION. A DATE is formatted according to the syntax of the dateTime datatype from the XML Schema specification that is based on ISO 8601.
Any other company-specific data can be stored in special data groups (SDGs).
EXAMPLE Company data

```
<COMPANY-DATA ID="SampleCompany">
       <SHORT-NAME>SampleCompany</SHORT-NAME>
       <LONG-NAME>Sample Company</LONG-NAME>
       <TEAM-MEMBERS>
              <TEAM-MEMBER ID="JDo">
                     <SHORT-NAME>JDo</SHORT-NAME>
                     <LONG-NAME>John Doe</LONG-NAME>
                     <DEPARTMENT>Development</DEPARTMENT>
                     <ADDRESS>5th Avenue</ADDRESS>
                     <ZIP>55555</ZIP>
                     <CITY>Big Apple</CITY>
                     <PHONE>+1 555 1234 567</PHONE>
                     <EMAIL>John.Doe@sample.com</EMAIL>
              </TEAM-MEMBER>
              <TEAM-MEMBER ID="JSm">
                     <SHORT-NAME>JSm</SHORT-NAME>
                     <LONG-NAME>John Smith</LONG-NAME>
                     <DEPARTMENT>Service</DEPARTMENT>
                     <ADDRESS>6th Avenue</ADDRESS>
                     <ZIP>55555</ZIP>
                     <CITY>Big Apple</CITY>
                     <PHONE>+1 555 1234 678</PHONE>
                     <EMAIL>John.Smith@sample.com</EMAIL>
              </TEAM-MEMBER>
       </TEAM-MEMBERS>
</COMPANY-DATA>
```

NOTE COMPANY-DATA and TEAM-MEMBER may be referenced by ADMIN-DATA. Therefore, a unique SHORT-NAME must be chosen for any of these objects to enable tools to offer a selection of employees and companies.

Usage of ADMIN-DATA

Revision information of ODX objects

The ADMIN-DATA of an ODX object may contain revision information specific to this object. Generally, we distinguish two types of revision information: Public revision information can be used by all process partners while internal revision information is provided for company-internal usage only. ADMIN-DATA supports the storage of both types of revision information.
When a new revision of an ODX object is created this may be documented by adding a new DOC-REVISION to the ADMIN-DATA of the object. The list of all DOC-REVISIONs of an ODX object represents the revision history for this object.
The mandatory DATE of the DOC-REVISION denotes the creation date of this revision. A DOC-REVISION contains both public and internal revision information. Public revision information may include a REVISION-LABEL to identify the revision, the STATE of the revision, the TOOL the revision was created with, the MODIFICATIONs made since the last revision, and the TEAM-MEMBER who created the revision. To make use of the DOC-REVISION's REVISION-LABEL the process partners have to agree on a common versioning scheme. The interpretation of STATE, TOOL, and MODIFICATIONs is also process-specific. Therefore, it's not covered by the ODX specification in further detail.
Each process partner can add internal revision information to a DOC-REVISION with a COMPANY-REVISION-INFO. The mandatory COMPANY-DATA-REF is used to identify the COMPANY-REVISION-INFO's owner, i.e. the process partner who added the internal revision information. This allows the process partners to store their internal revision information without interfering with each other. Optionally, a REVISION-LABEL and the STATE of the revision may be added. Such an internal REVISION-LABEL can be provided e.g. by a configuration management system that applies a proprietary versioning scheme.

Any COMPANY-REVISION-INFO is only relevant to its owner and should be left untouched by the other process partners.

NOTE As ODX is not supposed to define guidelines for version control and configuration management an runtime system is not required to process any revision information associated with ODX objects. Nevertheless, revision information of ODX objects can be used by specialized tools, e.g. for the look-up of a specific revision of an ODX object referenced by another ODX object; the REVISION attribute of the ODX link contains the (process-specific) description which revision of the link target is requested.

Updating the revision history of an ODX object

If a new revision of an ODX object is created then the revision history of any ODX object that contains the changed object has to be updated, i.e. a new revision of the containing object is added to its revision history. As an example, a new revision of a DIAG-SERVICE results in a new revision of the diagnostic layer instance the service instance belongs to. The DATE of the new DOC-REVISION of the containing ODX object should be set to a reasonable default value, e.g. the most recent DATE found in the DOC-REVISIONs of the contained ODX objects. How the REVISION-LABEL and other revision information is set is process-specific.

Versioning conflicts

When more than one process partner provide a new revision of the same ODX object at the same time this possibly results in a versioning conflict. To avoid this kind of versioning conflicts the process partners have to negotiate who is allowed to change an ODX object and its associated ADMIN-DATA. This way, at most one new revision of each ODX object is available at the same time.

7.1.3 VALUE CODING

In general values of type number in the ODX data must be decimal coded if not another number coding is defined by model or specification.

7.2 ODX PACKAGE

7.2.1 OVERVIEW

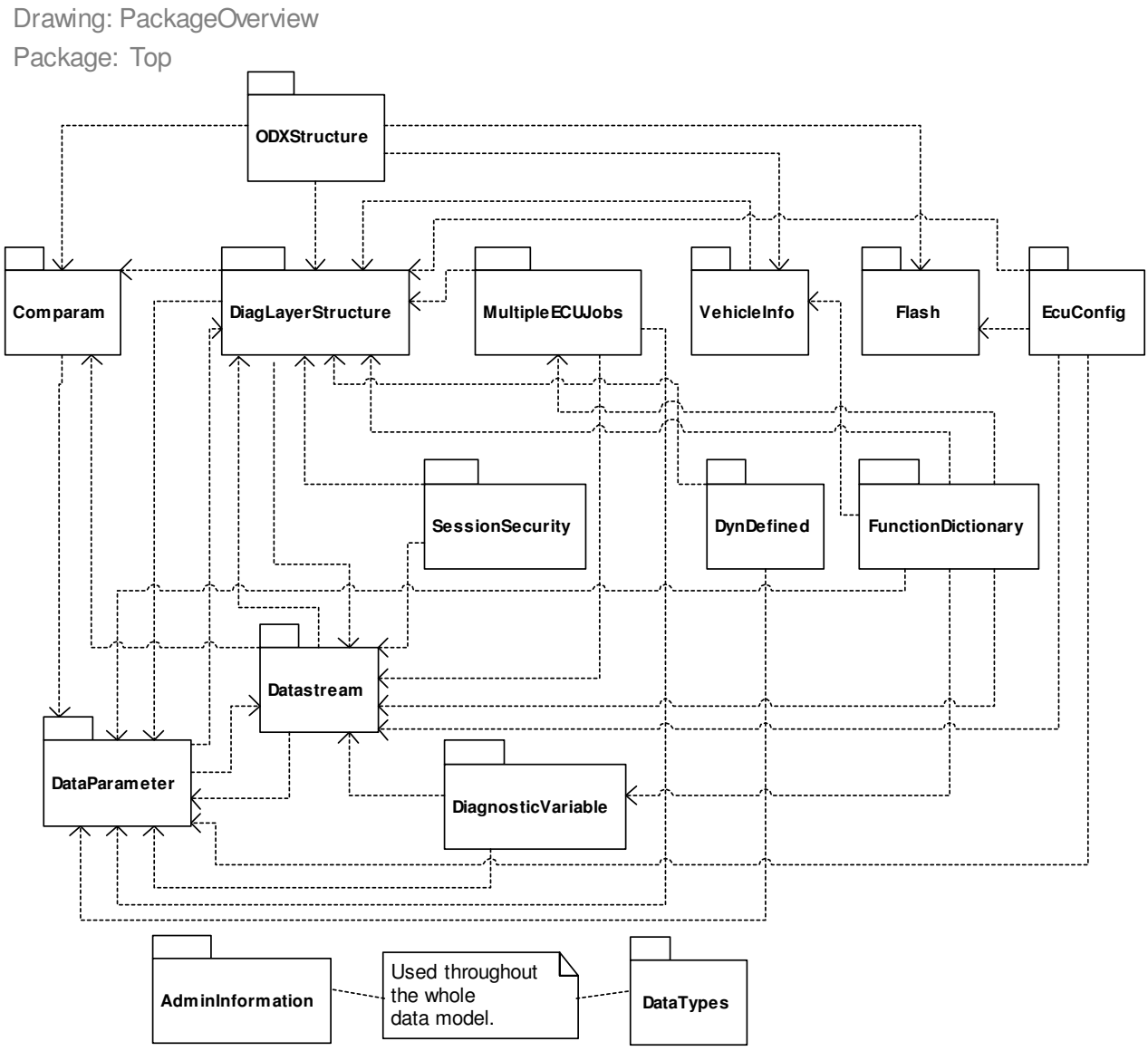


Figure 22 — Package overview

For easier understanding the UML model of ODX is divided into packages, which mostly reflect the logical structure of the model and to improve readability. The packages and their relations are shown in Figure 22. From the viewpoint of the data model ODXStructure is some kind of root package. In the following a short overview about the technical contents of the packages is given.

The package "ODXStructure" contains the very top-level root data of an ODX data instance, which splits into one of the following alternatives (see Figure 23):

i) One or a set of DIAG-LAYER structures i.e. logical layers for the description of diagnostic services with all necessary data (Package "DiagLayerStructure").

j) A COMPARAM-SPEC defines a specific protocol stack consisting of predefined communication parameter sets (COMPARAM-SUBSETs - specification of communication parameters, e.g. timings for physical layer, transport layer and diagnostic protocol). The comumunication parameters and those values are predefined in these COMPARAM-SUBSETs. The value of communication parameters may be changed in context of a layer, specific diagnostic service, LOGICAL-LINK or PHYSICAL-VEHICLE-LINK. This part of the model can also contain OEM specific parameters (Package "Comparam").

k) The definition of diagnostic jobs that deal with quasi-parallel communication with several ECUs (MULTIPLE-ECU-JOBS defined in the package "MultipleECUJobs").

l) Information about the access to a specific ECU in a vehicle network using gateways and using the concept of logical links between tester and a specific (physical) ECU (VEHICLE-INFO-SPEC defined in package VehicleInfo). More about vehicle information can be found in chapter 7.3.10.

m) The description of data which is needed for ECU flash memory programming like memory layout, logical structure of flash fragments and information (references) about the diagnostic services or jobs that have to be used to flash the data (FLASH defined in package "Flash").

n) An ECU's behaviour is strongly depending on the explicit configuration of a certain car. To reduce the amount of ECU-VARIANTS, ECU Configuration is performed. This section describes several usecases and the corresponding data model features.

o) Many functions of a car are distributed across several ECUs. The section FUNCTION-DICTIONARY describes the features of the ODX data model to cover the usecase of function-oriented diagnostics.

These data often descend from different sources and have their own and sometimes independent lifecycle. The information of DIAG-LAYER structures for example can be generated by the ECU supplier who on the other hand probably will not have complete information about the vehicle network where the ECU is to be integrated. This information is available at the OEM, which illustrates that for process reasons this data should be kept in separate parts. This on the other hand does not mean that objects in the ODX files are not linked at all. For the usage of the flash data for example the diagnostic jobs (or services) can be found in a DIAG-LAYER structure and are strongly linked to the information in the FLASH package.
An ODX instance may contain only one of the ODX-categories.

7.2.2 ODX STRUCTURE

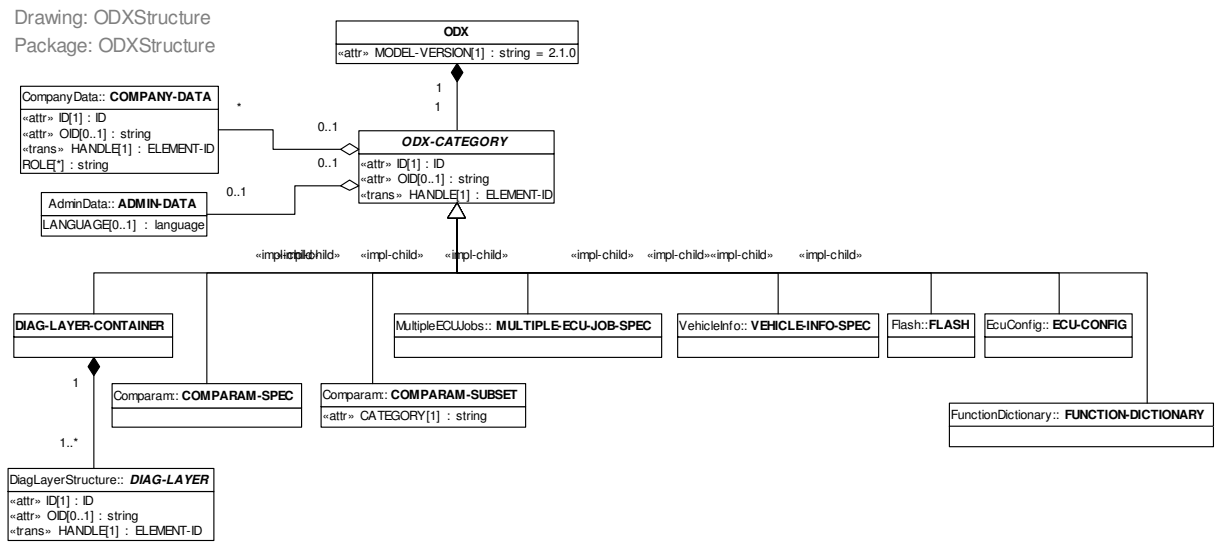


Figure 23 — ODX structure

A DIAG-LAYER has to be wrapped in a DIAG-LAYER-CONTAINER before it is transferred to the process partner. A DIAG-LAYER-CONTAINER may contain one or more DIAG-LAYERs and additional process-specific information like ADMIN-DATA and COMPANY-DATA. The process partner who receives the DIAG-LAYER-CONTAINER can extract the DIAG-LAYERs to store them individually; alternatively, the complete DIAG-LAYER-CONTAINER can be handled as a single storage unit.
DiagLayerStructure, Comparam, MultipleECUJobSpec, VehicleInfo, Flash

See previous section.
COMPARAM-SUBSET, ECU-CONFG, FUNCTION-DICTIONARY-SPEC

See previous section.
SessionSecurity
This package describes a data model for a state diagram, allowing to specify an ECU's diagnostic sessions, security states and corresponding methods to switch diagnostic sessions and to perform a security access.
DataParameter

This package deals with the description of simple (scalar) and complex (structured) data objects in the diagnostic data stream and with the conversion to symbolic values (name, value, unit). The important concept of DOP (data object property) is also defined in this package.
DataStream

This package deals with the description of the communication between diagnostic tester and ECU on "bits and bytes" i.e. diagnostic message or signal level. Diagnostic services and diagnostic jobs are defined as well as request and response telegrams (from a tester's point of view).
DynDefined

With some protocols (e.g. ISO 14230 or ISO-14229-1) it is possible to dynamically define data records at run time, which contain values from other (already existing) records. This package contains information used by services to define and to read the new data records.

DiagnosticVariable

This package describes an optional, alternative access to the ODX data, which is not communication oriented but rather quantity oriented. In the first case, the diagnostic application (or user) requires the tester to communicate with the ECU using a specific service (e.g. "read by local ID 44"). In the second case the user does not care about communication details at all but rather requests a specific variable from the ECU (e.g. "read the current value of Temperature T2").

AdminInformation

This package contains commonly used structures used to support the development process of diagnostic data like versioning, reference to content management systems and support of multi company projects.

DataTypes

This package contains the datatypes used in other packages and serves as a reference only.

7.3 ODX DATA MODEL FOR DIAGNOSTICS

7.3.1 OVERVIEW

The ODX standard is intended to contain all information that is required by a diagnostic tester to do diagnostic communication with a specific ECU or set of ECUs. This means that all ECU specific parameters for communication are completely contained in ODX and no ECU specific software in the tester software is necessary. This makes the tester completely data driven: When testing new ECUs no software must be updated, just the appropriate ODX data must be loaded.

The data description is designed to be independent of communication protocols, but - of course - ISO 14230 was often used to check the concepts of ODX data description practically. In order to reduce protocol specific data in the ECU description a special place for the definition of the set of protocol specific communication parameters was created (see 7.3.3 and 7.3.4).

Due to its intention to describe the diagnostic communication between tester and ECU the understanding of the data model part of the DiagLayerContainer naturally starts with looking at the description of the diagnostic communication services (DIAG-SERVICE). That means that a tester application that uses the ODX XML data will find ODX data – in a reduced view - to be a set of diagnostic services (i.e. the description of messages or frames on the communication bus). This corresponds to the concept of the object oriented interface MCD 3D / MVCI Diagnostic Server API where objects diagnostic service and job play a key role.

This does not mean that the DIAG-SERVICE is the only "entry point" of the ODX data. With respect to the use case of measurement and data acquisition the concepts of DIAG-VARIABLES (see 7.3.7) and TABLE (see 7.3.6.11) are an alternative way to use the ODX data. This means that you can look up the ODX XML data for a required measurement variable without caring about the underlying communication services that are used to obtain the current value of an internal variable of the ECU.

It is recommended to read the details of the DIAG-LAYER data model starting from the DIAG-SERVICE class. Speaking not of classes but of objects, if you pick out a DIAG-

SERVICE object you get the root of a tree of related objects (aggregation or reference) that are used to describe the parameters that are necessary for a runtime system to do the diagnostic communication. All objects not linked in this chain (or tree) with DIAG-SERVICE being the root object are - from the view of a runtime system - not of relevance for communication. If for example a POS-RESPONSE (description of a communication telegram from the ECU) is defined which is not used (referenced) by any DIAG-SERVICE it will never become effective i.e. it cannot be used for communication (GLOBAL-NEG-RESPONSEs building a noteable exception). However, other elements like TABLE or DIAG-VARIABLE can be used as an initial access to the data through an MCD 3D / MVCI server API.

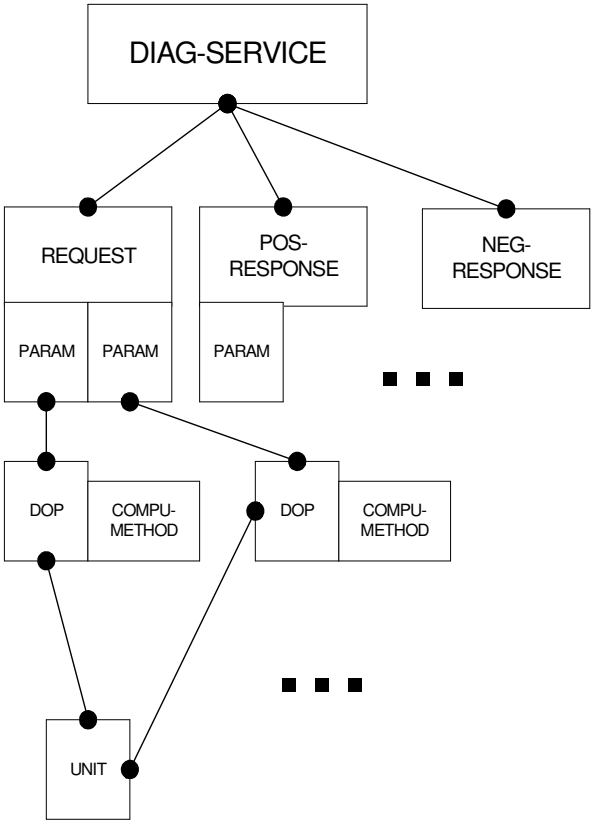


Figure 24 — Object chain in ODX DiagLayerStructure

In all cases where a re-usage of objects is possible these objects are connected to other objects via references ("links" in XML). Since these links sometimes must be created interactively with an editor tool, normally all objects that can be reused by reference do have a characteristic set of attributes: SHORT-NAME (unique identifier for this specific class of object), LONG-NAME and DESC.
Only a set of DIAG-SERVICEs is visible from the viewpoint of an application using MCD 3D / MVCI Diagnostic Server API to access the ODX data - slightly simplified. As it can be seen later, this set of DIAG-SERVICEs depends on the "location" that is used in the ODX data model. Example: The number of supported services depends on the variant of an ECU. If you for example enter the ODX data model at the location of the "EU" (Europe) variant of an ECU, you may get a different set of DIAG-SERVICEs than compared e.g. with another variant like e.g. "JP" (Japan). It is of course possible to have only one set of DIAG-SERVICEs in an ODX file, but for the benefit of avoiding redundant data it is possible to spread the DIAG-SERVICEs over several hierarchical layers of the data model as shown in section 7.3.2. The layers are top down: PROTOCOL, FUNCTIONAL-GROUP,

BASE-VARIANT, ECU-VARIANT. ECU-SHARED-DATA is a special pool of information as described later in section 7.3.2.

7.3.2 DIAGNOSTIC LAYER STRUCTURE

7.3.2.1 OVERVIEW

The ODX data model structures the diagnostic data in five so-called diagnostic layers, which pursue the following purposes:

- Implement a hierarchical model to achieve data abstraction across a set of ECU variations, protocols, functional groups and libraries in order to limit data redundancy. This special (limited) form of inheritance is being called value inheritance in this document. This includes the possibility to overwrite certain objects that are contained in a more general layer by another object defined in a more specialized layer.
- Provide a library-like mechanism via ECU-SHARED-DATA
- Create a framework that supports ECU variant identification and base variant identification
- Reflect the needs of the MCD 3D / MVCI Diagnostic Server API by defining the set of objects that are visible at the MCD 3D / MVCI Diagnostic Server API. The objects being controlled that way are instances of the following classes (Note that the concept of visibility is only defined for these three categories):

— DIAG-SERVICE

— SINGLE-ECU-JOB

— DIAG-VARIABLE

— TABLE

- Define the scope for the ODX-link and short-name referencing mechanisms (see 7.3.13)

Figure 25 shows a simple example of how the various diagnostic layers can be used to implement:

- A protocol (ISO15765)
- Two base variants (a body control module BCM and a door control module DCM)
- Two variants of the BCM and one variant of the DCM
- A library to collect globally defined PIDs (parameter identifiers)
- A functional group definition to allow functional communication to the complete DOORS system (functional addressing)

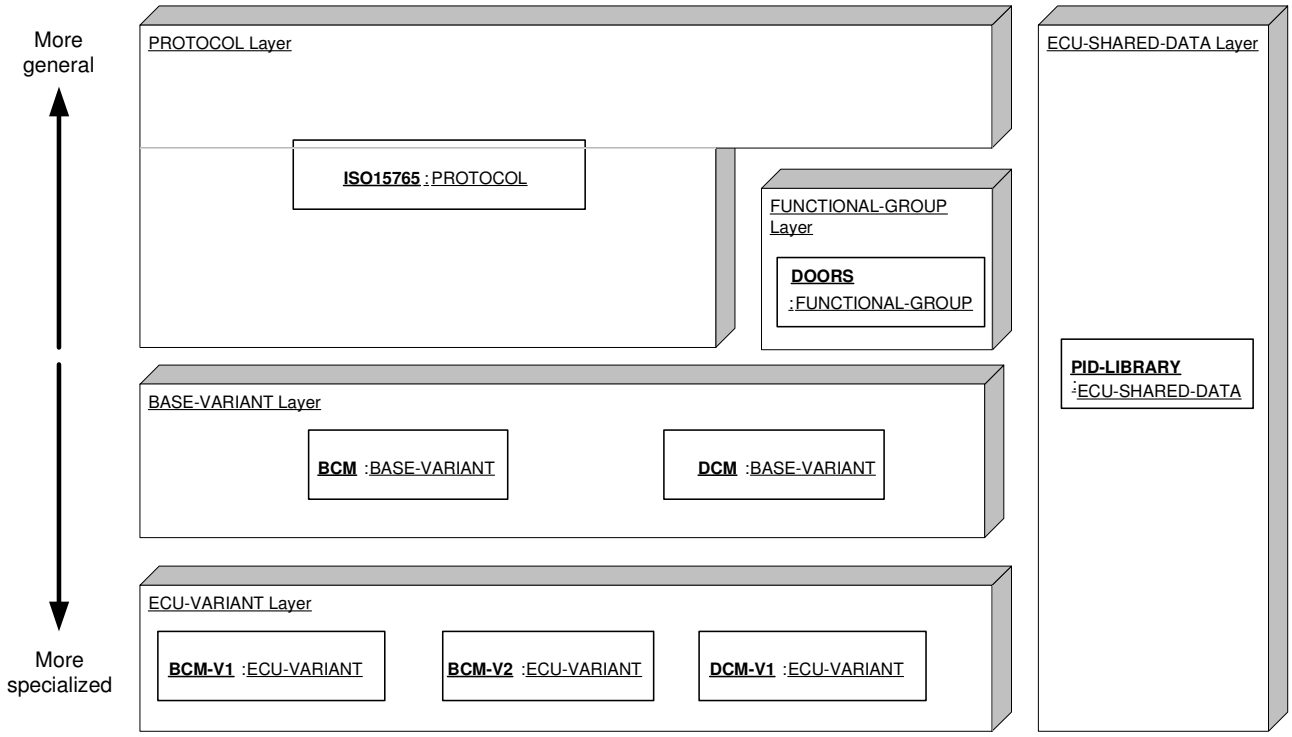


Figure 25 — Diagnostic layers overview example

7.3.2.2 DIAGNOSTIC LAYER MODELLING

The various diagnostic layers have many common characteristics that have been modelled using the common base class DIAG-LAYER. The classes PROTOCOL, FUNCTIONAL-GROUP, BASE-VARIANT and ECU-VARIANT are the building blocks of the hierarchy mentioned in the previous section. The ECU-SHARED-DATA class is somewhat different, which will be elaborated later.
A list of POSSIBLE-PHYSICAL-LINK-TYPEs allows to restrict the set of physical vehicle links which can be used in conjunction with this PROTOCOL.

Drawing: DiagLayers
Package: DiagLayerStructure

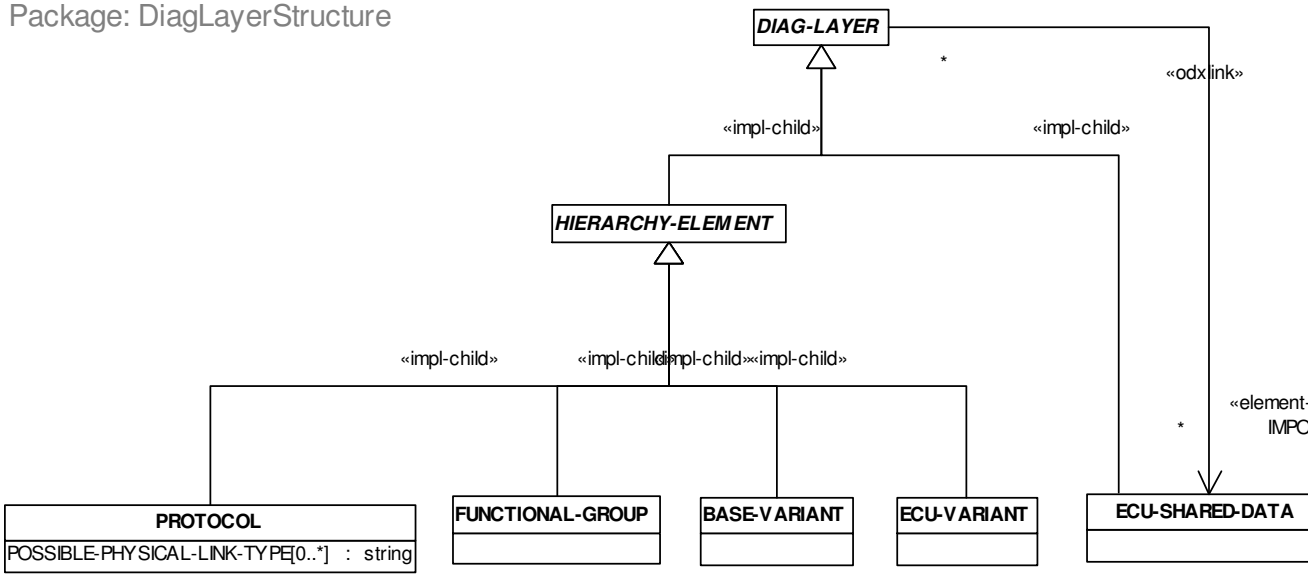


Figure 26 — Diagnostic layers

7.3.2.3 DIAGNOSTIC LAYER COMMONALITIES

The model section shown in Figure 27 defines the common characteristics for all diagnostic layers. The base class DIAG-LAYER aggregates the following components:

- Collection of REQUEST objects (see 7.3.5.5)
- Collection of response objects (POS-RESPONSE, NEG-RESPONSE, GLOBAL-NEG-RESPONSE) (see 7.3.5.6)
- Document information using the ADMIN-DATA and COMPANY-DATA objects (see 7.1.2.3
- Collection of FUNCT-CLASS objects
  A FUNCT-CLASS object is being defined at the DIAG-LAYER and is used to categorize the DIAG-COMM objects that belong to the DIAG-LAYER. The semantics of the created categories are user specific and not defined by this document
- Optional DIAG-DATA-DICTIONARY-SPEC object
  Even though this element is optional, most DIAG-LAYER instances will contain it, because it serves as the main container for the data elements necessary to decode the diagnostic messages.
- Collection of DIAG-COMM objects and/or DIAG-COMM references (bundled through a DIAG-COMM-PROXY in the model – such a PROXY is not visible in an instance file)
  The DIAG-COMM class is an abstraction of DIAG-SERVICE and SINGLE-ECU-JOB (see 7.3.5.2), which are exposed at the MCD 3D / MVCI Diagnostic Server API. The set of DIAG-COMM objects and DIAG-COMM references that is given here, defines what is visible at the MCD 3D / MVCI Diagnostic Server API for the corresponding DIAG-LAYER.

— Collection of ADDITIONAL-AUDIENCEs as described in section 7.1.2.2

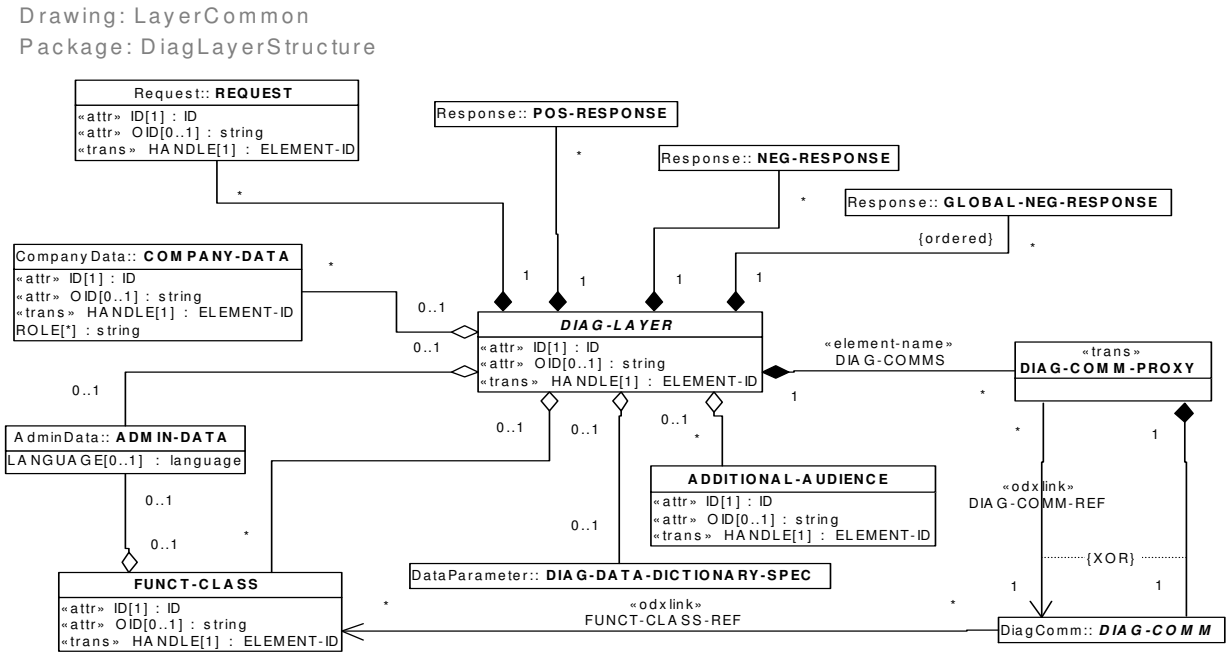


Figure 27 — Diagnostic layer commonalities

7.3.2.4 VALUE INHERITANCE

Value inheritance is a core concept of ODX. Value inheritance makes the inheritance concept of object-oriented technology usable for diagnostic data modelling. The benefits of value inheritance are

— Reuse of diagnostic data for more than one ECU or ECU-variant based on a single source principle
— Reduce the amount of data by specifying additions to and differences between ECU projects, rather than repeating data used in more than one ECU project
— Provide data security and integrity
— Avoid error-prone copies of identical data between ECU projects.

7.3.2.4.1 General

Value inheritance is a relationship between diagnostic layers, i.e. the classes PROTOCOL, FUNCTIONAL-GROUP, BASE-VARIANT, ECU-VARIANT and ECU-SHARED-DATA. Value inheritance means that data objects, which are contained within a DIAG-LAYER **A** are considered to be also contained in a DIAG-LAYER **B** that establishes

a value inheritance relationship to **A**. Value inheritance is implemented by references in the inheriting layer to the inherited layer(s).
A diagnostic layer of a specific type (PROTOCOL, FUNCTIONAL-GROUP, BASE-VARIANT, ECU-VARIANT or ECU-SHARED-DATA) can inherit only a specific set of other types of diagnostic layers. E.g. a diagnostic layer cannot inherit an other diagnostic layer of the same type. In this way, an inheritance hierarchy is established between the diagnostic layer classes. In Figure 28 — Diagnostic layer hierarchy illustrated the hierarchy.

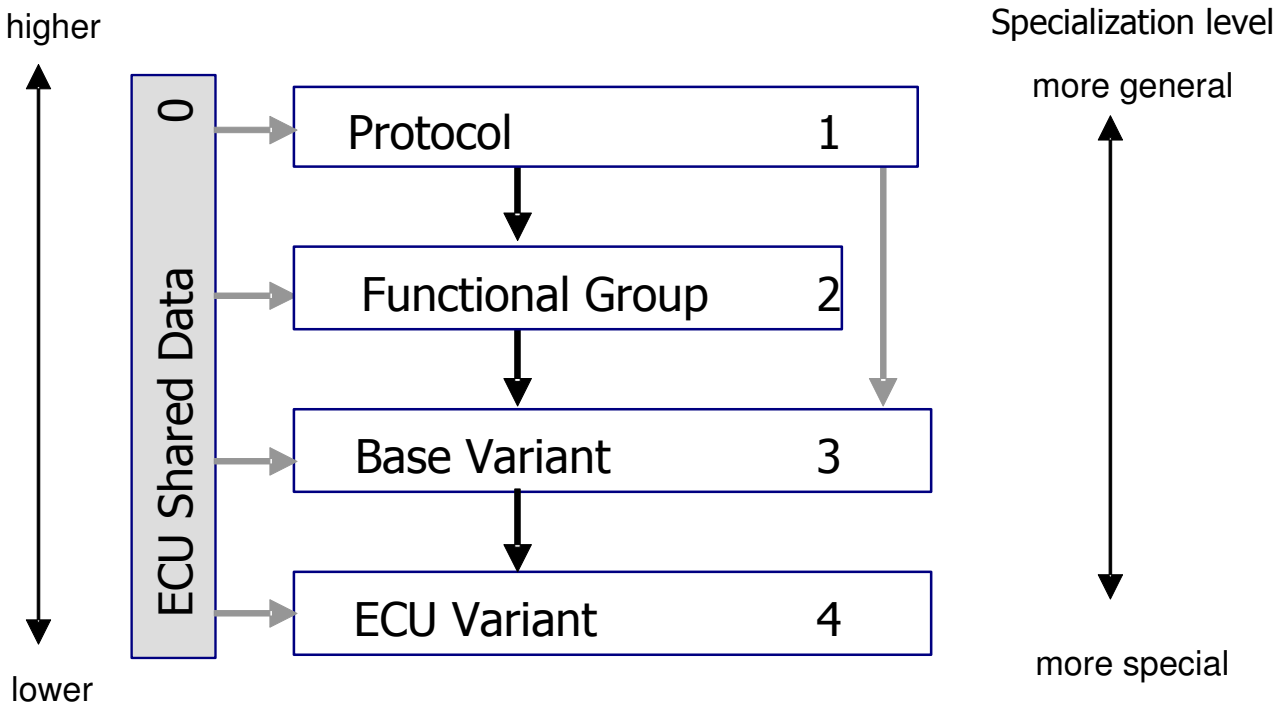


Figure 28 — Diagnostic layer hierarchy

Diagnostic layers higher in hierarchy are termed ‘more general’, layers lower in hierarchy are termed ‘more special’.

Figure 29 —UML representation of the diagnostic layer hierarchy is the UML model of the inheritance hierarchy. Table 1 is a tabular representation of the same information.

The following restrictions apply to this general definition of Value inheritance:

— Only objects having a HANDLE attribute can be value-inherited.

— If an object having a HANDLE attribute aggregates other objects that also have HANDLE attributes, the complete aggregation is being inherited. I.e. always the outermost object of an aggregation (which has a HANDLE attribute) is subject of value inheritance.

— Objects of the following classes are subject to value inheritance:

- COMPANY-DATA
- FUNCT-CLASS
- DIAG-COMM (its specialisations)

- RESPONSE
- REQUEST
- TABLE
- DOP-BASE (its specialisations)
- UNIT
- PHYSICAL-DIMENSION
- UNIT-GROUP
- DIAG-VARIABLE
- VARIABLE-GROUP

As a result, these data objects (or data objects contained within them) can now be used as targets for either odx-link or short-name references. Another possibility to enlarge the data pool from which data objects may be referenced is given with the import mechanism (see 7.3.2.5).

NOTE There are more classes within the ODX data model that have a SHORT-NAME and that are subject of references to their SHORT-NAME. However, only the classes mentioned within the list above are subject to value inheritance.

Value inheritance is implemented by the use of special odx-link references named PARENT-REF as shown in Figure 30 — Parent reference. A DIAG-LAYER refers to other DIAG-LAYERs using such an odx-link to establish a value-inheritance relationshipFigure 29 —UML representation of the diagnostic layer hierarchy shows the possible relationships, which can be constructed between DIAG-LAYERs using the PARENT-REF specialisations.

EXAMPLE A BASE-VARIANT object establishes a value-inheritance relationship to a PROTOCOL object by aggregating a PROTOCOL-REF object that in turn references the PROTOCOL object by using an odx-link.

Between a pair of DIAG-LAYER objects may exist either a value inheritance relationship as described in this section or an import relationship as described in 7.3.2.5. The existence of both kinds of relationships for the same pair of objects is prohibited.

rawing:LayerHierarchy
ackage:DiagLayerStructure

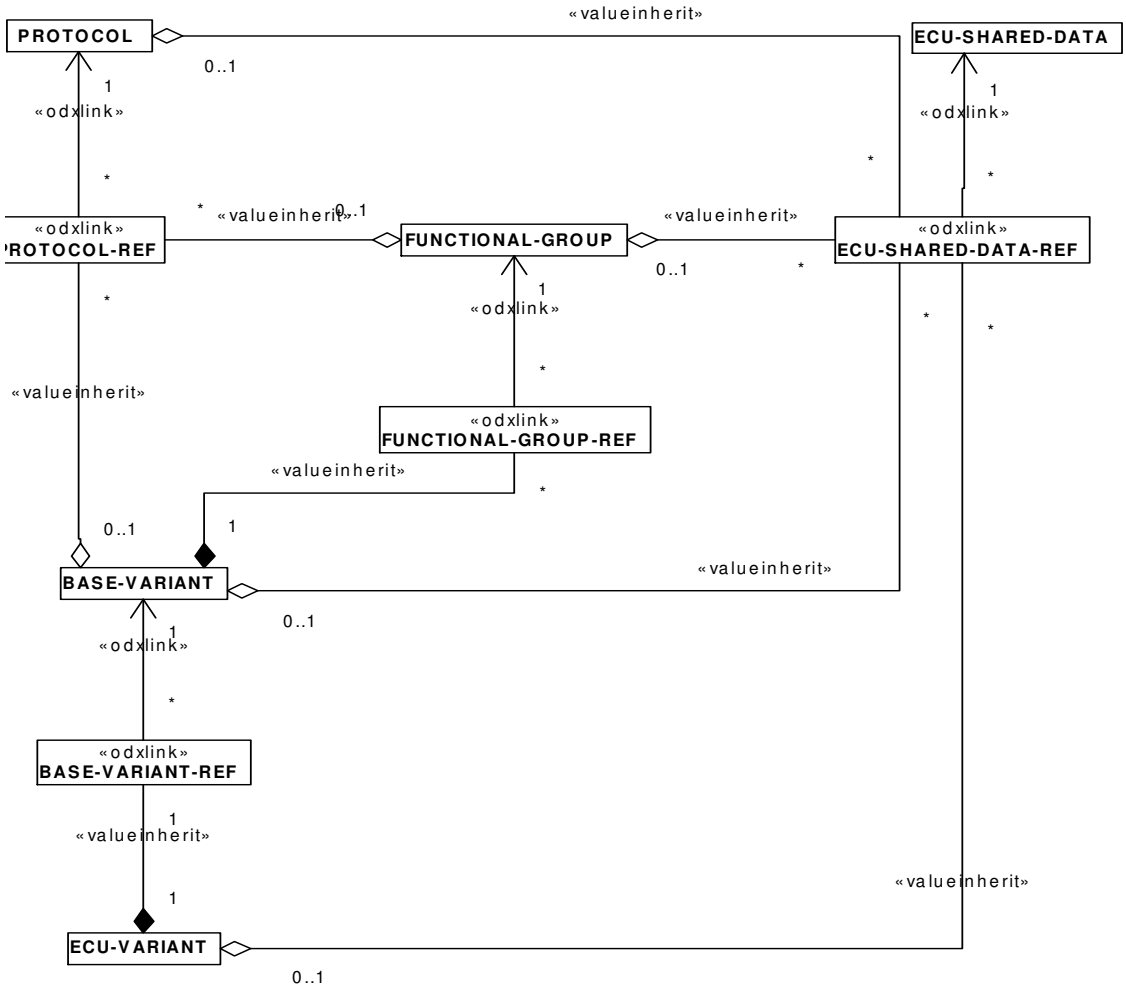


Figure 29 —UML representation of the diagnostic layer hierarchy

The following table shows the possible value-inheritance relationships among the various DIAG-LAYER specialisations.

Table 1 — Value inheritance

DIAG-LAYER	May value-inherit from				
	PROTOCOL	BASE-VARIANT	ECU-VARIANT	ECU-SHARED-DATA	FUNCTIONAL-GROUP
PROTOCOL	No	No	No	Multiple	No
BASE-VARIANT	Multiple	No	No	Multiple	Multiple
ECU-VARIANT	No	Exactly one	No	Multiple	No
ECU-SHARED-DATA	No	No	No	No	No
FUNCTIONAL-GROUP	Multiple	No	No	Multiple	No

7.3.2.4.2 Overwriting of objects

As mentioned above, the value inheritance concept includes the possibility to overwrite objects that have been inherited from a more general layer. This is being accomplished by matching the SHORT-NAME of the object to be overwritten. There are distinct namespaces within which the SHORT-NAME resolution takes place. The implemented class hierarchy within the ODX model implicitly gives these namespaces. I.e. all specialisations of a certain base class share the same namespace for their SHORT-NAMEs. 7.3.13 gives a comprehensive overview about the namespaces.
If the overwritten object is among those objects that are visible at the MCD 3D / MVCI Diagnostic Server API (see 7.3.2.4.4), the object defined in the more specialized layer will be visible.
All SHORT-NAME references (either contained in objects inherited from more general layers or defined within the current layer) are targeting the object defined in the more specialized layer. This implements the "late-binding" mechanism of ODX. Objects that are imported from an ECU-SHARED-DATA layer acquire the level of specialisation from the inheriting layer. Objects that are value-inherited from an ECU-SHARED-DATA layer obtain a specialisation level that is less specialized than the inheriting layer but more specialized than the next more general layer within the layer hierarchy.
If an ECU-VARIANT value-inherits DIAG-SERVICES with the same SHORT-NAME from the BASE-VARIANT as well as from an ECU-SHARED-DATA the DIAG-SERVICE from the ECU-SHARED-DATA is supposed to overwrite the one from the BASE-VARIANT.
If a DIAG-LAYER value-inherits a DIAG-SERVICE from an ECU-SHARED-DATA with the same SHORT-NAME as a DIAG-SERVICE it defines locally, the locally defined DIAG-SERVICE will overwrite the value-inherited one Objects that are imported from an ECU-SHARED-DATA layer aquire the level of specialisation from the inheriting layer. Objects that are value-inherited from an ECU-SHARED-DATA layer obtain a specialisation level that is less specialized than the inheriting layer but more specialized than the next more general layer within the layer hierarchy.
If e.g. an ECU-VARIANT value-inherits DIAG-SERVICES with the same SHORT-NAME from the BASE-VARIANT as well as from an ECU-SHARED-DATA the DIAG-SERVICE from the ECU-SHARED-DATA is supposed to overwrite the one from the BASE-VARIANT.
If e.g. a DIAG-LAYER value-inherits a DIAG-SERVICE from an ECU-SHARED-DATA with the same SHORT-NAME as a DIAG-SERVICE it defines locally, the locally defined DIAG-SERVICE will overwrite the value-inherited one.

7.3.2.4.3 Multiple inheritance

Because the ODX data model allows multiple value inheritance, the following situations may occur:

— A diagnostic layer inherits multiple different data objects having the same SHORT-NAME by establishing value inheritance relationships to multiple other diagnostic layers

This is prohibited but can be resolved by creating NOT-INHERITED-branches for all but one of the conflicting data objects. These objects have to be created within the more specialised diagnostic layer.
This method of resolving multiple inheritance conflicts exists for data objects within NOT-INHERITED-branches. In all other cases, value inheritance relationships that would cause multiple data objects having the same SHORT-NAME at a certain specialisation level must not be established.

— A diagnostic layer inherits the same data object through multiple inheritance paths

EXAMPLE: Each of two PROTOCOL instances **A** and **B** establish a value inheritance relationship to an ECU-SHARED-DATA instance **C**. All data objects contained within **C** virtually exist now within **A** and **B** at the same level of

specialisation. If now a BASE-VARIANT **D** establishes value inheritance relationships to **A** and **B** it seems to get a duplicated set of **C**'s data objects.
This shall be resolved in a way that only a single set of **C**'s data objects exists within **D**. This can be accomplished using the ID of data objects.

7.3.2.4.4 Visibility at MCD 3D / MVCI Diagnostic Server API

One purposes of value-inheritance is to reflect the needs of the MCD 3D / MVCI Diagnostic Server API by defining the set of objects that are visible at the MCD 3D / MVCI Diagnostic Server API. All of the objects that are defined within the network formed through value-inheritance relationships are visible at the MCD 3D / MVCI Diagnostic Server API. To provide further control about what objects are visible, an exclusion mechanism has been established by means of the NOT-INHERITED-branches as shown in the figure below.

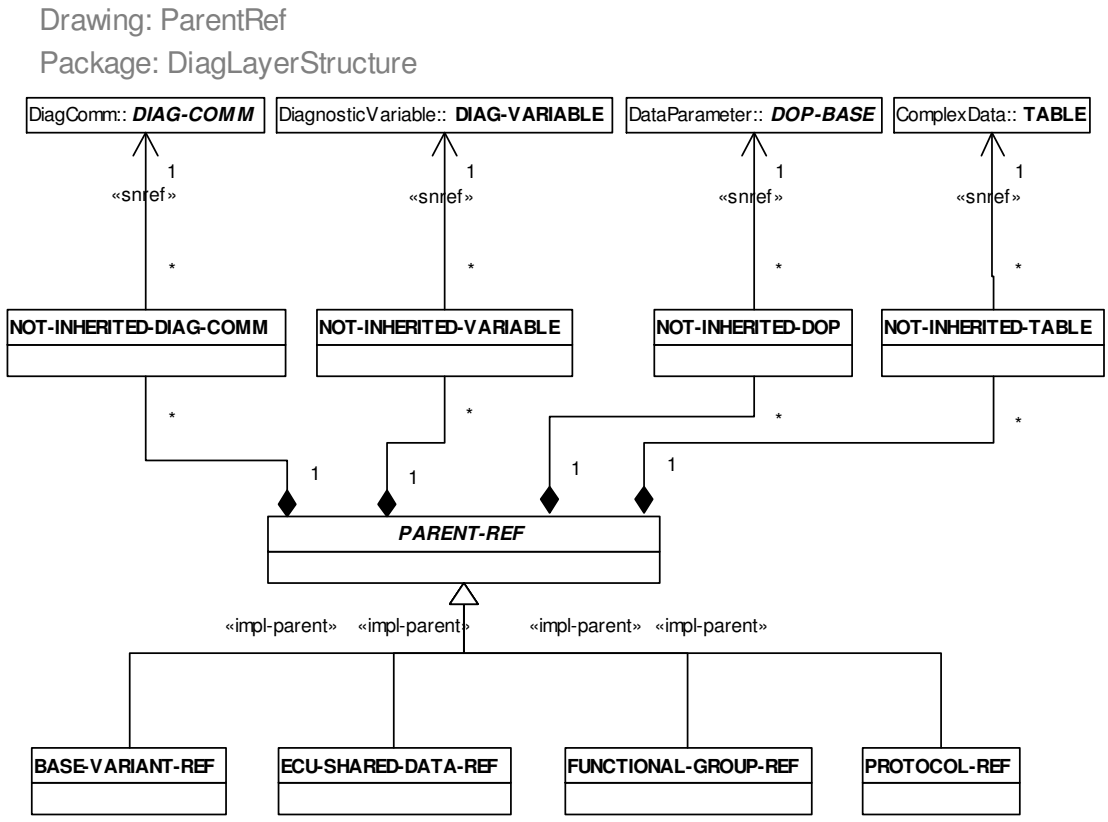


Figure 30 — Parent reference

7.3.2.4.5 Value Inheritance Examples

Some examples are provided to illustrate the value inheritance concept. A, B, C are DIAG-LAYERS and objects O, defined in a specific layer are drawn in black. Layers further down in the diagram inherit from the layer just above, indicated by an open arrow. Objects inherited are drawn in grey. Each example exhibits four columns: Column 1 identifies the layers and the objects defined in each layer. Dashed arrows are used to illustrate references (odxlinks as well as SNREFS). Column 2 shows how inheritance and/or overwriting of objects will impact referencing. Column 3 lists those objects, that a

MVCI-Server will return, if asked to list the objects of the respective layer. Column 4 lists all objects usable for communication by a LOGICAL-LINK on the respective layer.
An object O1' represents an object of the identical type (class) and identical SHORT-NAME to object O1.
Note the symbols in the following examples of this section:

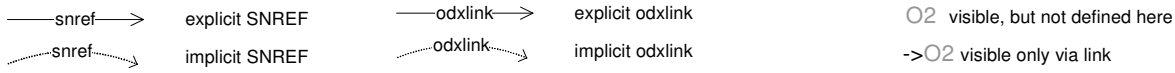
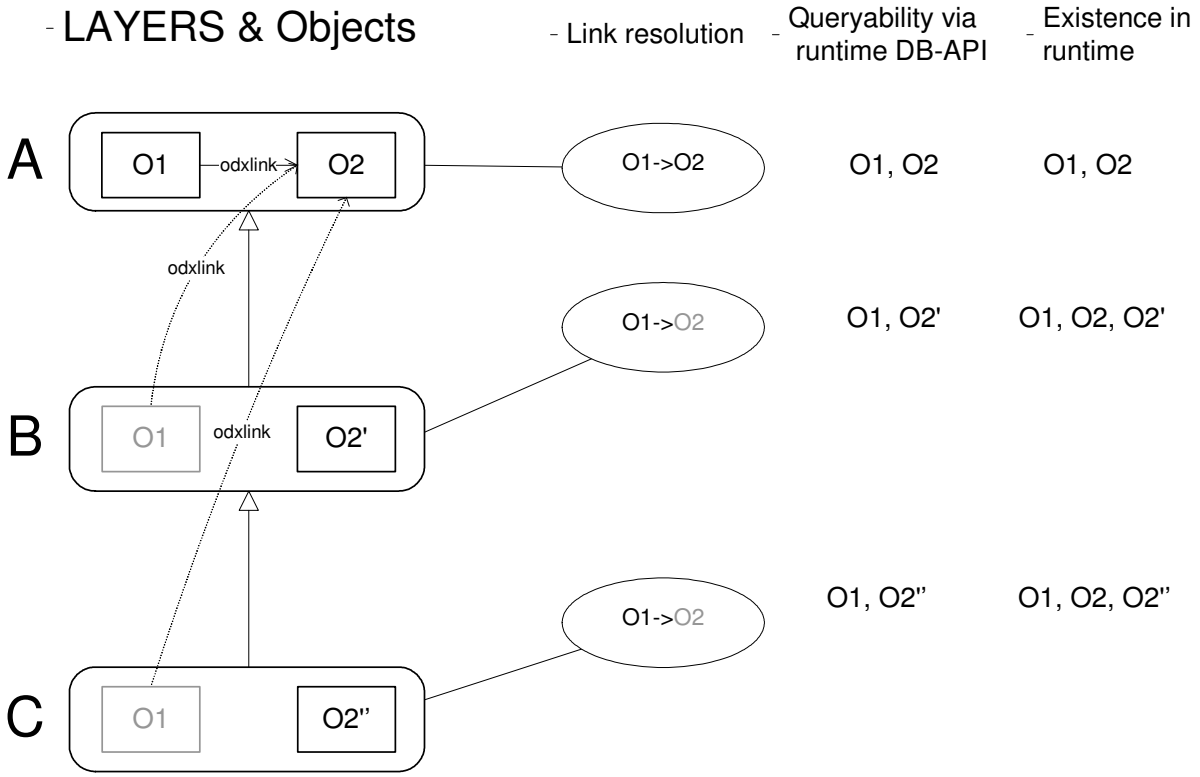


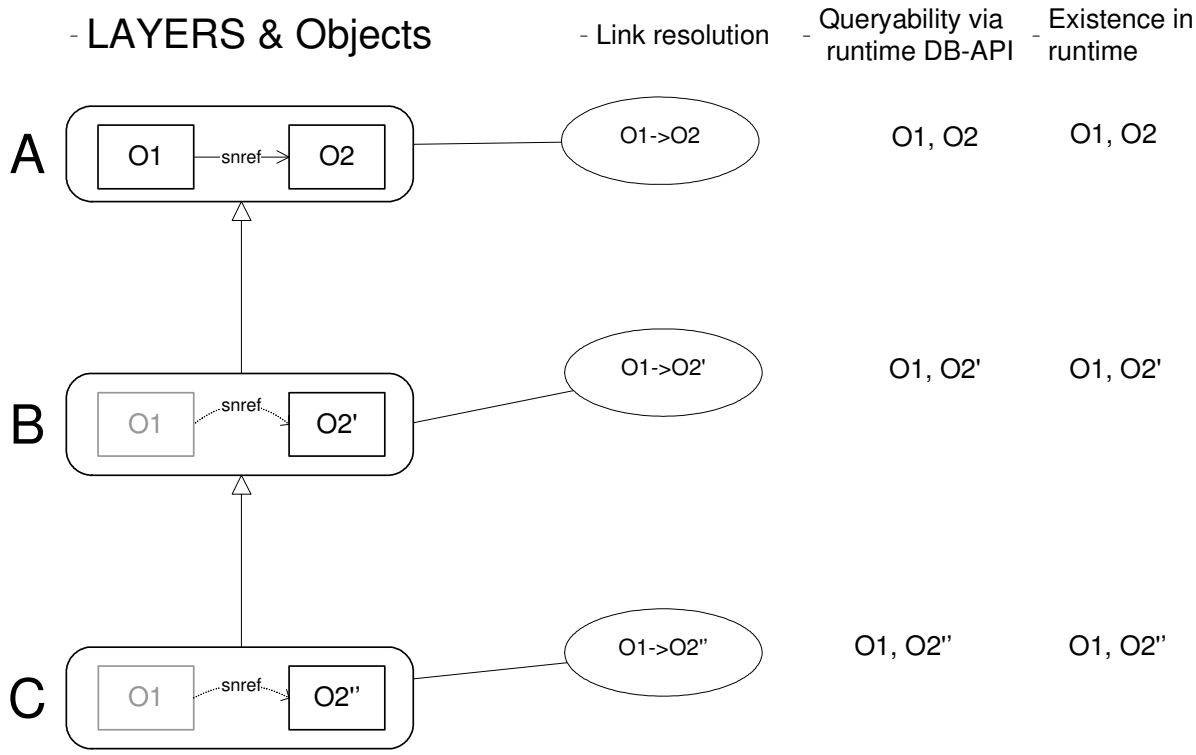
Figure 31 — Symbols for the value inheritance examples

Example 1 illustrated the case, where object O2, defined in layer A is successively overwritten on both inheriting layers B and C. Yet because the object O1 references O2 in A via odxlink, O2 remains to be visible and usable from within layers B and C. Example 2 shows a very similar situation, however, since SNREFS are used in place for the odxlinks, this impacts visibility and usability of object O2 on layers, lower in the inheritance hierarchy.



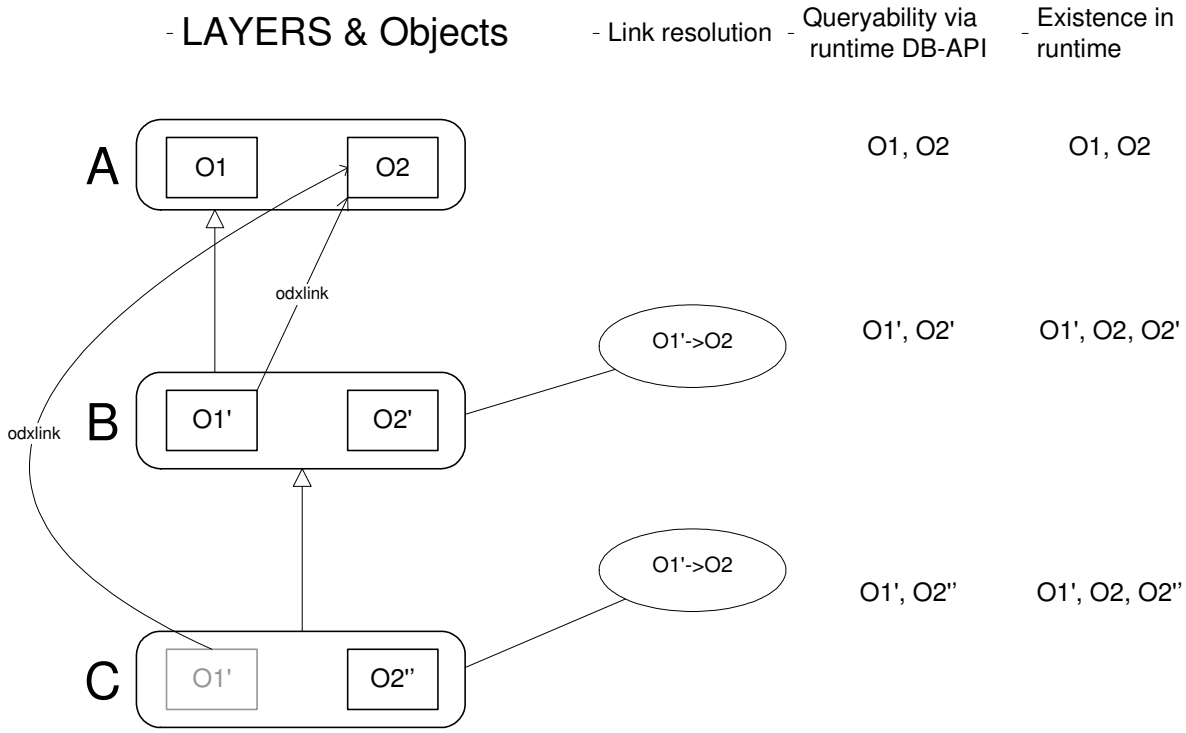
Example 1 — Inheritance and odx-link referencing

In example 2 references via SNREF refer to overwritten objects O2' and O3' respectively rather than to the original object from higher layers like odxlinks do. Objects overwritten on lower layers are no longer usable at runtime on the respective specialisation level, unless they are references via odxlink (like in example 1).



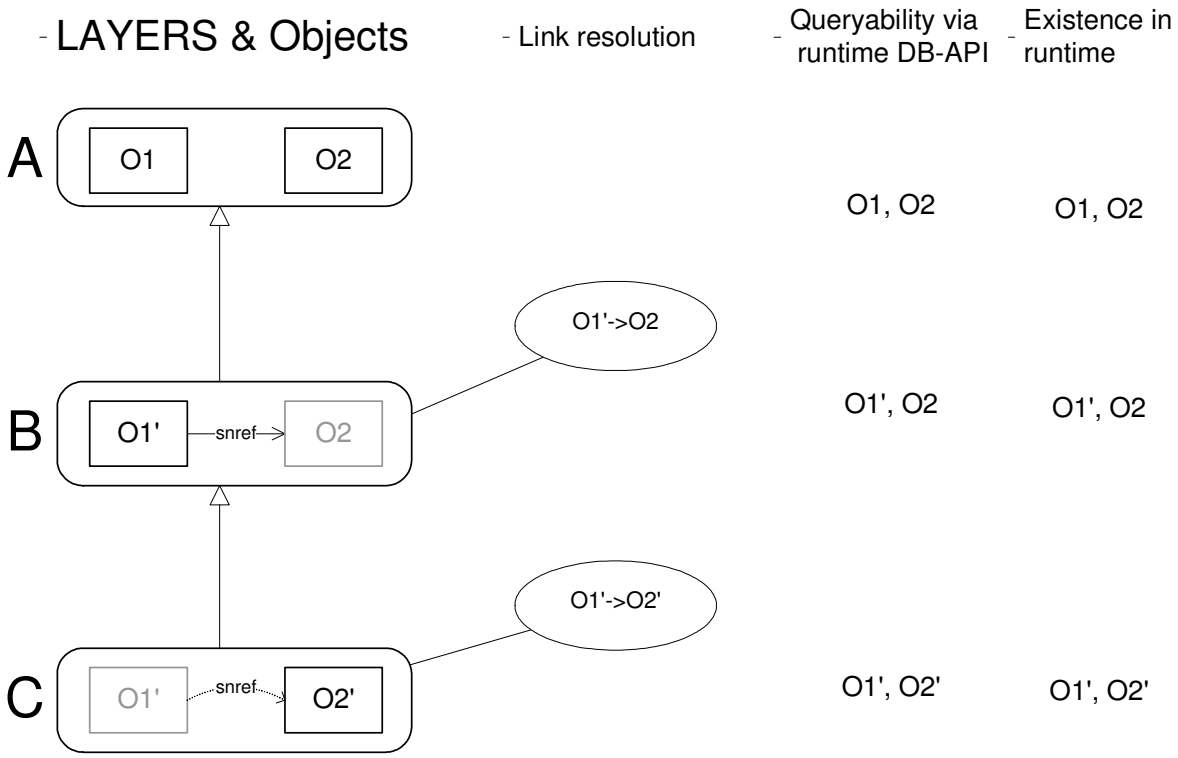
Example 2 — Inheritance and SNREFs

In example 3 the object O2 retains its usability on layers B and C even though it is overwritten, However it will only be visible as an object in layer A, not in B and C.



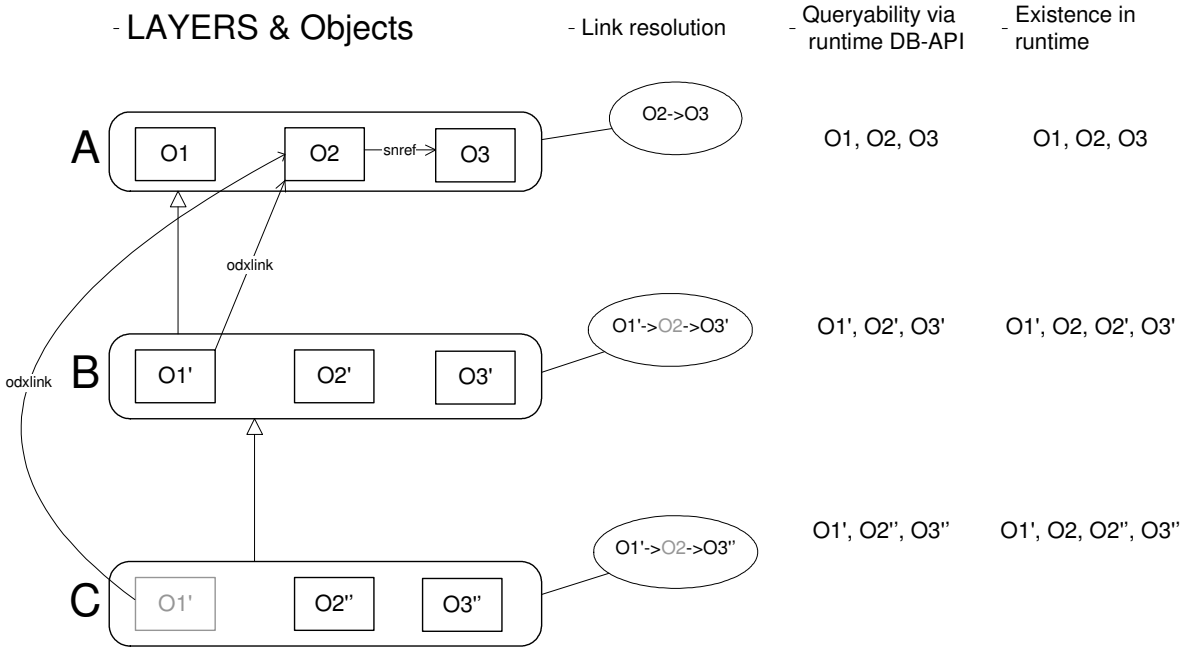
Example 3 — Visibility of overwritten objects

Example 4 illustrates a case where an object O2, that has SNREFs pointing to it, is overwritten by O2', SNREF to O2 will point to O2' on the layer where O2' is defined and on layers lower in the inheritance hierarchy.



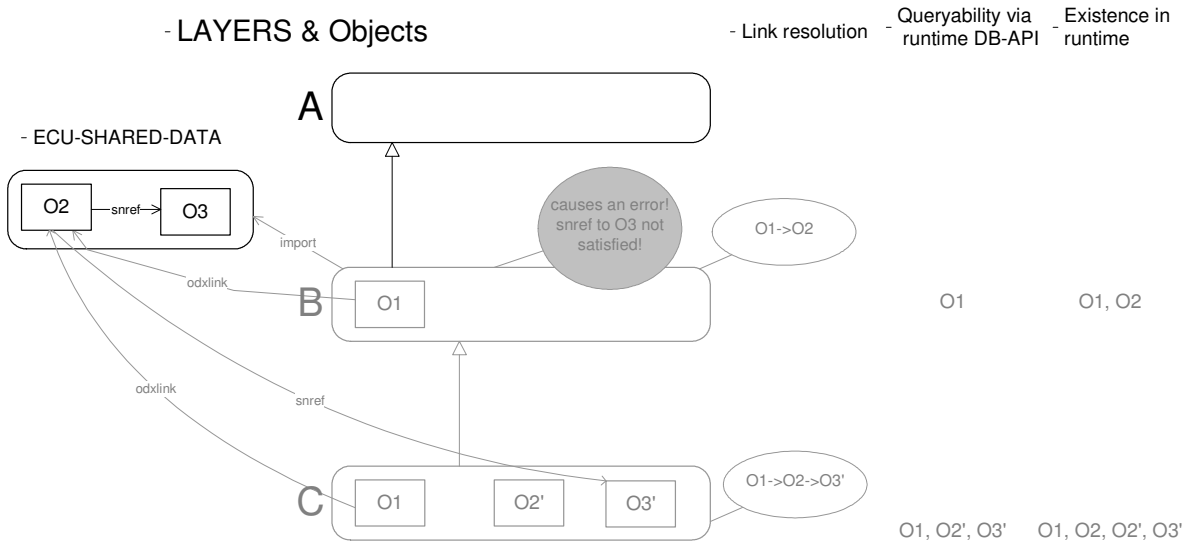
Example 4 — Overwriting objects referenced by SNREFs

Example 5: In contrast to SNREFs, odx-links preserve the identity of the objects they point to, e.g. O2, even if O2 is overwritten on lower layers.



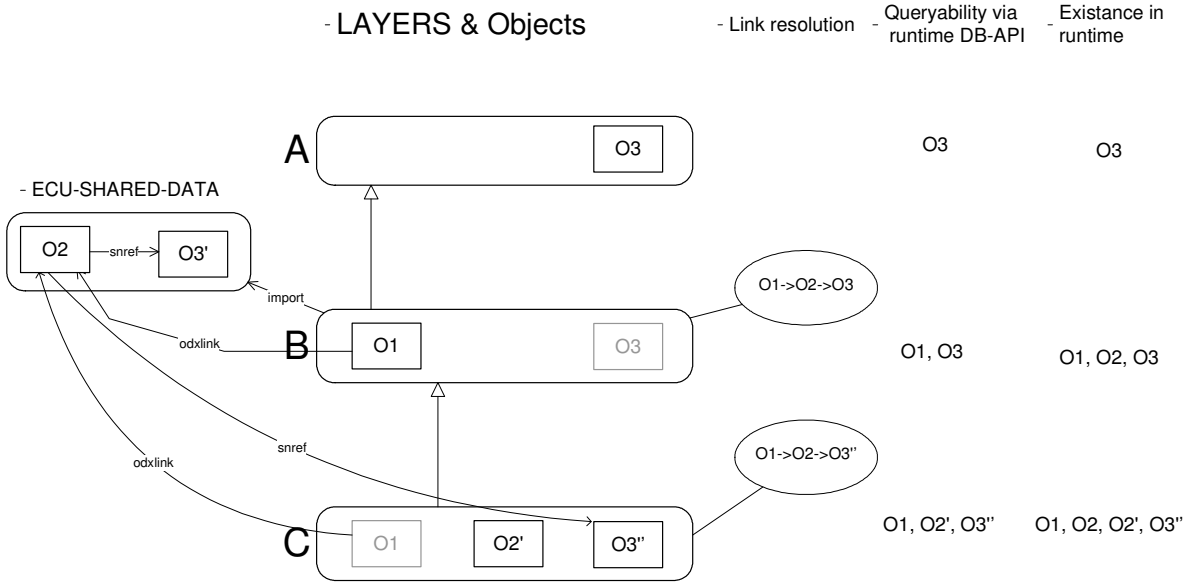
Example 5 — Overwriting objects referenced by odx-link

Example 6: In this example the SNREF from O2 to O3 in the ECU-SHARED-DATA can be resolved inside the ECU-SHARED DATA, however it cannot be resolved in layer B since O3 is not imported into B. This situation constitutes a configuration invalid in ODX and should be flagged as an error.



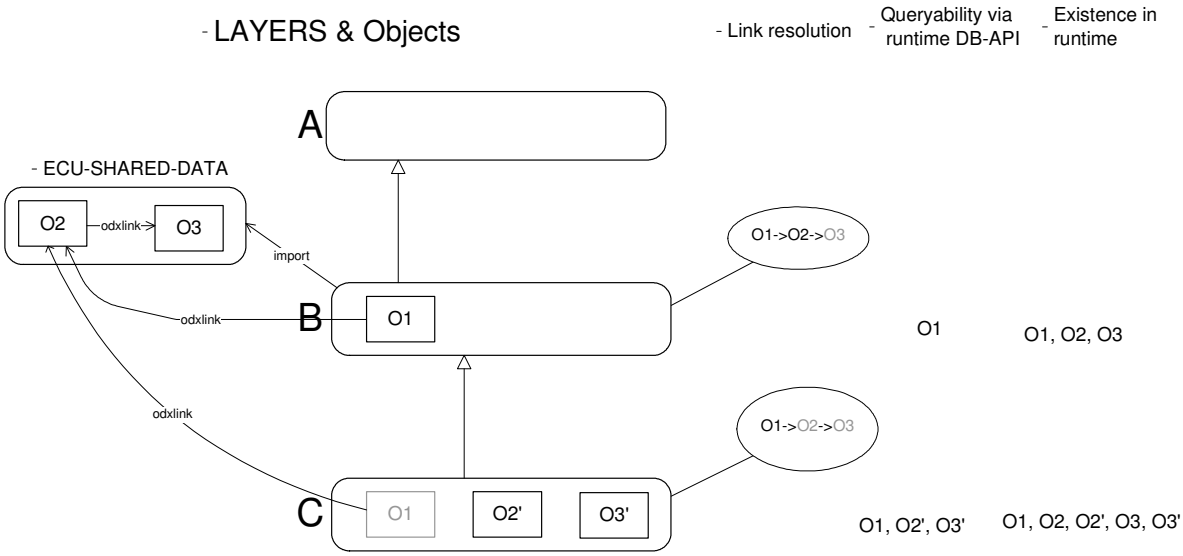
Example 6 — Referencing object in ECU-SHARED-DATA

Example 7: When there is an object O3 in B the SNREF between O2 and O3' in the ECU-SHARED-DATA can also be resolved in B. This example is valid in ODX.



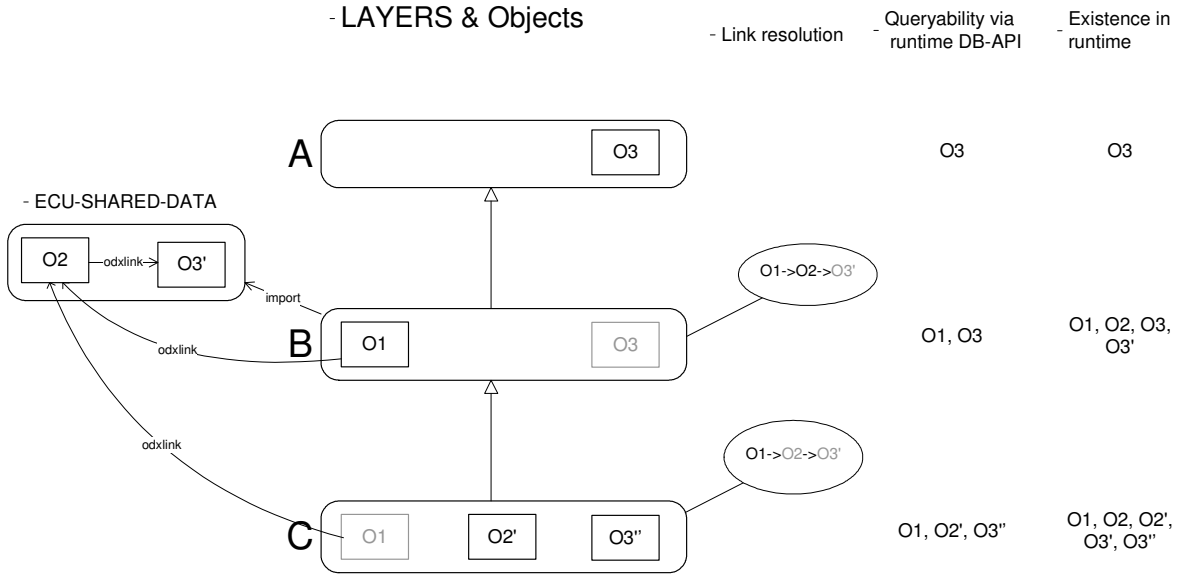
Example 7 — Overwriting object from ECU-SHARED-DATA in other layers

Example 8: Like in any other layer, odxlinks preserve the identity of the objects they point to. Overwriting odxlink targets affects the visibility of objects. Overwritten objects are replaced by the overwriting object at and below where they are defined in the inheritance tree.



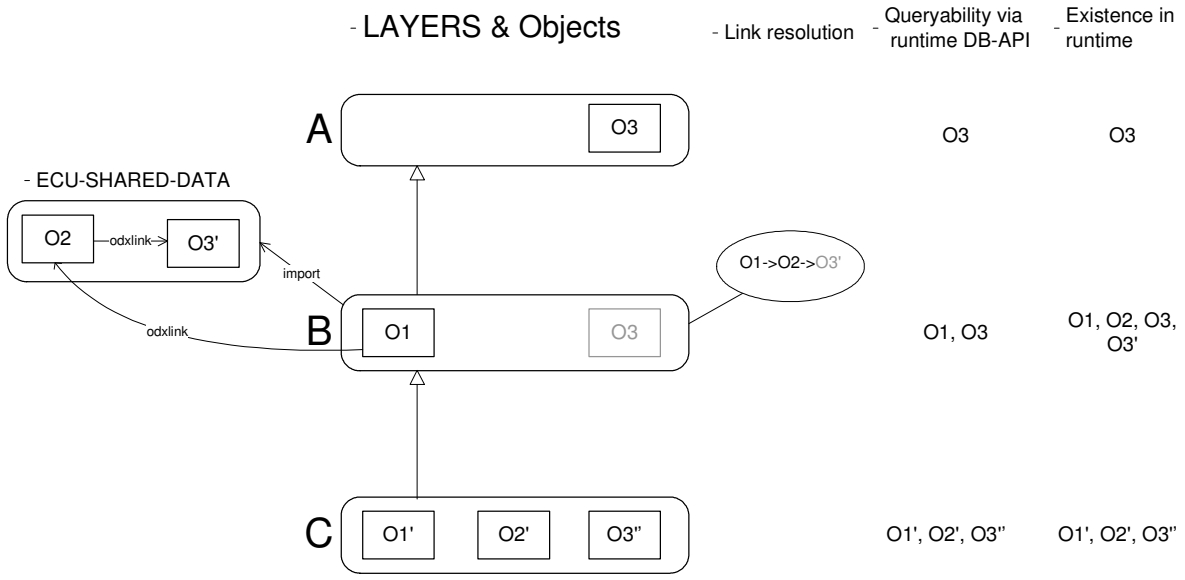
Example 8 — odxlinks inside ECU-SHARED-DATA

Example 9: This situation is identical to example 5. Overwriting an object O3' in an ECU-SHARED-DATA, that an odxlink points to will preserve the identity of the referenced object O3', whether it is defined in an ECU-SHARED-DATA or elsewhere.



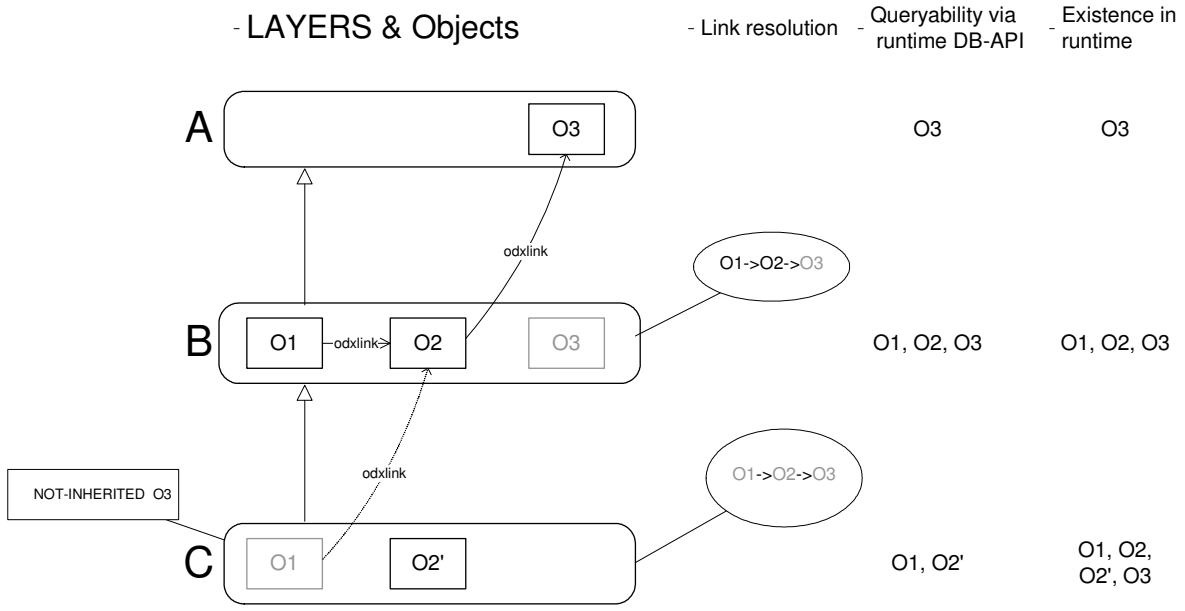
Example 9 — odxlinks inside ECU-SHARED-DATA to overwritten objects

Example 10: If an object O1 is overwritten by O1', the references of O1 are not automatically preserved in O1'.



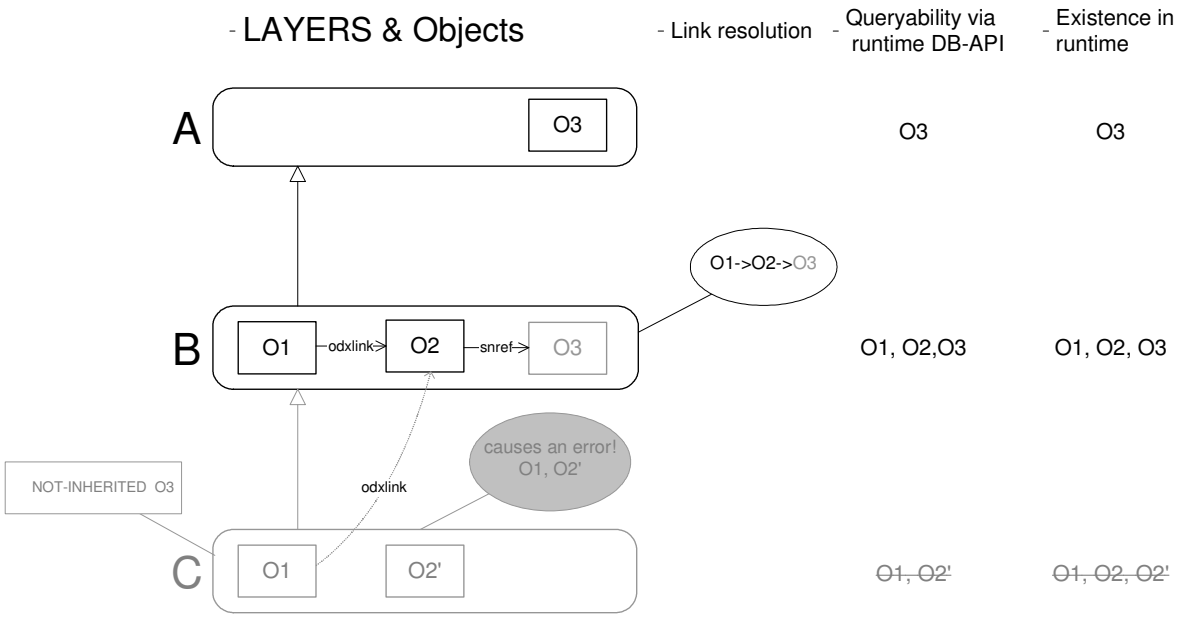
Example 10 — Overwriting objects with references

Example 11: An object O3, that has been excluded from inheritance remains usable in layer C if valid odxlinks are established to it, even though O3 is not listed when all objects of layer C are queried.



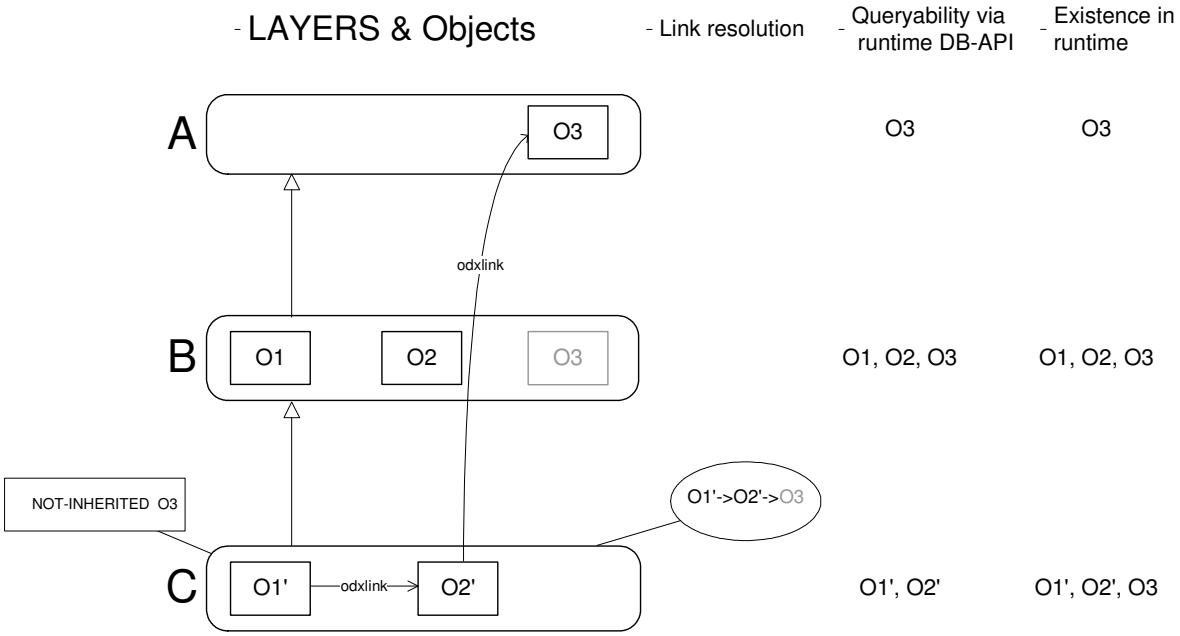
Example 11 — NOT-INHERITED mechanism and odxlinks

Example 12: Unlike odxlinks, SNREFs are valid only, when the referenced object is defined or inherited in the respective layer. Example 12 constitutes an invalid specification because O3 is not an object in layer C. This situation may be resolved by using an odxlink in place of the SNREF, c.f. example 13.



Example 12 — NOT-INHERITED mechanism and SNREFs

Example 13: Resolve reference resolving problem of example 12.



Example 13 — NOT-INHERITED mechanism, resolving an invalid SNREF by replacing it by an odxlink

7.3.2.5 IMPORT

Another method provided by the ODX data model to avoid redundant data is the use of ECU-SHARED-DATA layers, which fulfil the purpose of libraries. As already mentioned in 7.3.2.5, an ECU-SHARED-DATA object is very similar to the other DIAG-LAYERs with the exception that it can also be the target of an import relationship. If a DIAG-LAYER object imports an ECU-SHARED-DATA object A, it enlarges its data pool available to be

referenced by odx-link by the data objects contained within A. In order to make use of the imported data objects, they have to be referenced by odx-link. That means imported data objects cannot be targets of short-name references.

- All DIAG-LAYER specialisations can establish multiple import relationships to ECU-SHARED-DATA objects.
- Only ECU-SHARED-DATA can be a target of IMPORT-REF.
- ECU-VARIANT can import from another ECU-VARIANT having a value-inheritance relationship to the same BASE-VARIANT.

NOTE All objects of type ECU-VARIANT, which have a value-inheritance relationship to the same BASE-VARIANT object, form a data pool, from which all these ECU-VARIANTs may reference data objects using the odx-link mechanism. There is no additional import relationship between those ECU-VARIANT layers necessary.

An odx-link association as shown in Figure 26 is implementing the import mechanism. If a DIAG-LAYER **A** imports a data object with the same SHORT-NAME as a data object that is being value-inherited, the imported data object is treated as it would have been defined in **A** and therefore overwrites the inherited data object.

7.3.2.6 VARIANT IDENTIFICATION

The ECU-VARIANT specialisation of a DIAG-LAYER may also contain data to support variant identification at runtime. The figure below shows the data structures that are dedicated to that purpose. Please see section 7.4.3 for details.

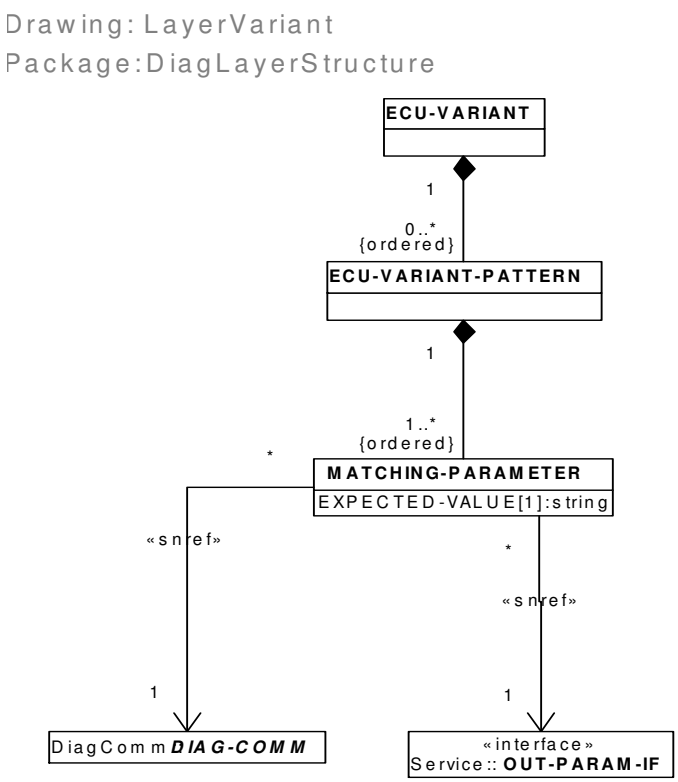


Figure 32 — Data structures for variant identification

7.3.2.7 BASE VARIANT IDENTIFICATION

The BASE-VARIANT-PATTERN element allows to specify how a base variant can be identified by communicating functionally or physically to an ECU. The figure below shows the data structures that are dedicated to that purpose. Please see section 7.4.4 for details on how to set up a base variant identification data structure.

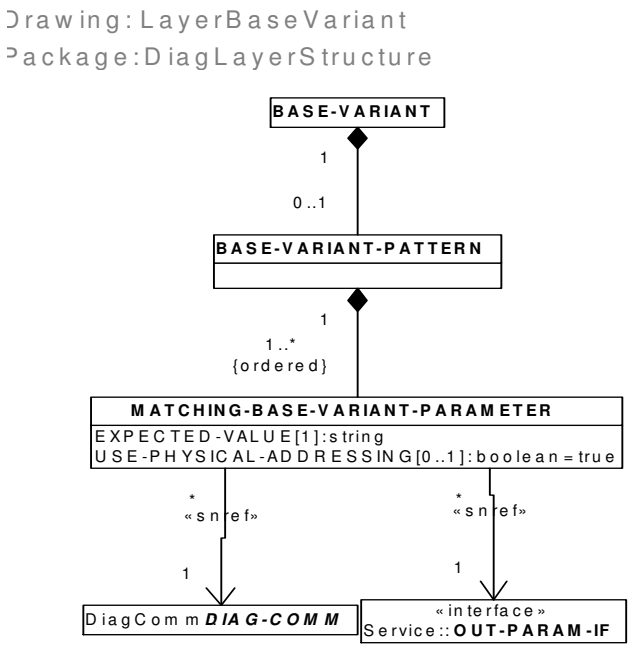


Figure 33 — BASE-VARIANT-PATTERN

7.3.3 COMMUNICATION PARAMETER

The timing and the logical behaviour of the runtime system is parameterized by communication parameters. These are represented in the ODX data by COMPARAM and COMPLEX-COMPARAM elements. For the definition of communication parameters with simple values (e.g. a P2 timing parameter, et.c) the COMPARAM element shall be used. Structured communication parameters consisting of multiple simple COMPARAMs (e.g. an array of CAN-IDs for functional addressing) must be definied using the COMPLEX-COMPARAM element. The ALLOW-MULTIPLE-VALUES attribute of a COMPLEX-COMPARAM allows to overwrite a COMPLEX-COMPARAM with multiple value structures within one DIAG-LAYER. This is needed, e.g. in the use case of functional addressing, where the e.g. the CAN-Ids of multiple responding ECUs have to be set up. The set of COMPARAMs and COMPLEX-COMMPARAMs usually do not change the data bytes of the telegram from a diagnostic tester to the ECU, defined elsewhere in the data model, but may manipulate parts of the telegram header. An example is the ECU address in the header, or the CAN identifier. Communication parameters can only be defined within a COMPARAM-SUBSET as part of a COMPARAM-SPEC structure.

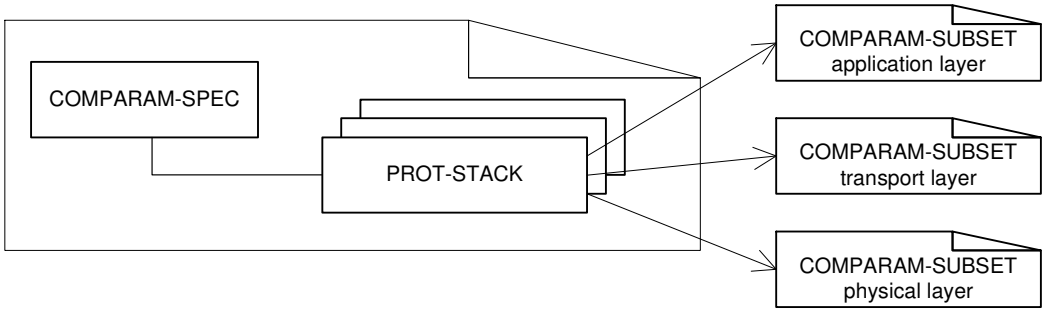


Figure 34 — Split up of the COMPARAM-SPEC structure (simplified)

A COMPARAM-SPEC structure consists of one or more sub-structures called PROT-STACK. A PROT-STACK contains a protocol specific set of predefined communication parameters. The parameter values may be changed in context of a DIAG-LAYER or specific diagnostic service. The attribute PDU-PROTOCOL-TYPE at PROT-STACK defines the standard protocol where the communication parameter set is designed for. The standard protocol must be well known to the protocol driver. This protocol type name should be a concatenation of the application layer specification name plus the transport layer specification name, connected by the additional string “_on_”. There exists a predefined value set of standardized protocol types (see Table A.6 — Enumeration of TYPE at PROTOCOL). The master reference for the list valid protocol type names is the D-PDU-API specification.
How the PROT-STACKs are grouped by the COMPARAM-SPEC structure is not defined by this document. It is recommended, however, to group all the PROT-STACK structures together into a COMPARAM-SPEC which belong to a combination of an application layer and a transport layer specification. The set of communication parameters defined by a PROT-STACK is subdivided into several COMPARAM-SUBSET structures according to the ISO OSI layer model. Usually there is one structure for each application layer, transport layer and physical layer. The attribute CATEGORY defines which layer a COMPARAM-SUBSET is designed for.
To ensure the exchangeability of ODX data, an ODX compliant tool must support the exchange of COMPARAM-SPEC structures. The runtime system identifies COMPARAMs and COMPLEX-COMPARAMs in the ODX data that matches the internally used communication parameters via their SHORT-NAME. As a consequence, the SHORT-NAME of the COMPARAMs and COMPLEX-COMPARAMs is standardized in ISO-22900-2, which are part of the standardized COMPARAM-SPEC structures which must not be changed. In case of the non-standardized (system-specific) COMPARAMs and COMPLEX-COMPARAMs with a different semantic to the standardized ones, the SHORT-NAMEs can be freely chosen. It causes, that two runtime systems can use the same (non-standardized) parameter with different names. The mapping of such parameters is not specified within this document.
The communication parameters defined in a COMPARAM-SPEC or rather their values can be changed outside the COMPARAM-SPEC structure by referencing the COMPARAM or COMPLEX-COMPARAM element via SHORT-NAME and defining a different value. This can be done using a COMPARAM-REF element. Changing a communication parameter value the BASE-VALUE of the specific parameter must be considered.
For a COMPARAM-REF to a COMPARAM the new value has to be definied as SIMPLE-VALUE.

```
<COMPARAM-REF ID-REF="CP_14291">
    <SIMPLE-VALUE>2001</SIMPLE-VALUE>
</COMPARAM-REF>
```

A COMPARAM-REF to a COMPLEX-COMPARAM requires, that all SIMPLE-VALUEs of the including COMPARAMs need to be covered by a COMPLEX-VALUE.

```
<COMPARAM-REF ID-REF="CCP_11381">
    <COMPLEX-VALUE>
        <SIMPLE-VALUE>4</SIMPLE-VALUE>
        <SIMPLE-VALUE>2018</SIMPLE-VALUE>
        <SIMPLE-VALUE>0</SIMPLE-VALUE>
    </COMPLEX-VALUE>
</COMPARAM-REF>
```

NOTE A COMPARAM-REF may also be directed at a COMPARAM that is specified within a COMPLEX-COMPARAM to only overwrite parts of a COMPLEX-COMPARAM's default values. In this case the COMPARAM-REF is used in the exact same manner as for a regular COMPARAM.

COMPARAM-REF elements can be attached at a DIAG-LAYER in case of changing a communication parameter value globally for all DIAG-COMMs of this diagnostic layer. The use of the COMPARAM-REF can be called "overwriting of a communication parameter value" because the communication parameter values are inherited from DIAG-LAYER to DIAG-LAYER (diagnostic layer). The inheritance of data between DIAG-LAYERs is usually called "value-inheritance" within this document. As already mentioned, a PROT-STACK contains a protocol specific set of predefined communication parameters. This indicates that at any time during runtime exactly one PROT-STACK is active. The valid PROT-STACK at a time is referenced via short-name by the active PROTOCOL or by the active LOGICAL-LINK. If there is a reference defined from the active PROTOCOL and also from the LOGICAL-LINK they must be identical.

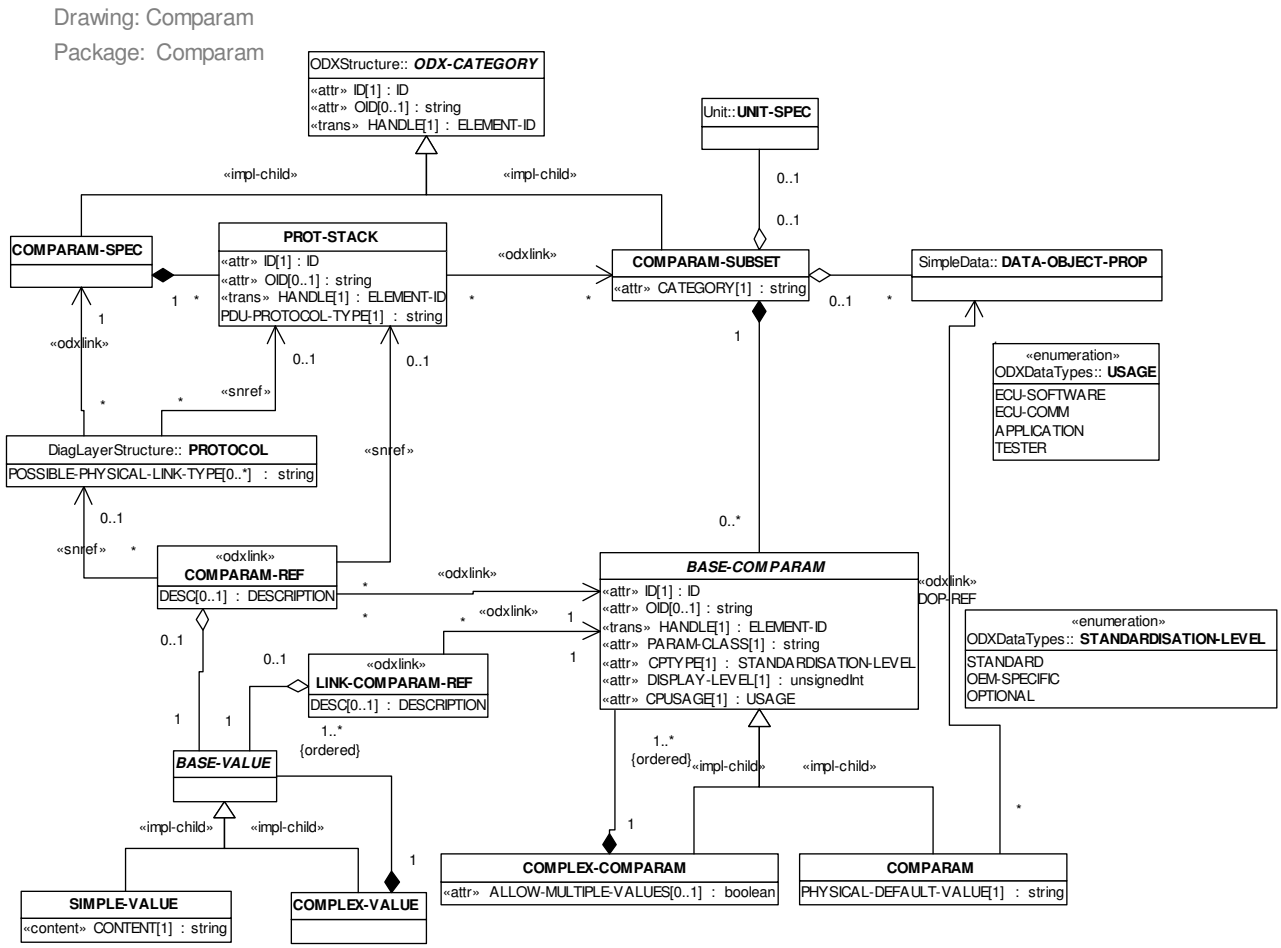


Figure 35 — Communication parameters

Each COMPARAM has a name (SHORT-NAME), a physical default value (PHYSICAL-DEFAULT), and a reference to a data object property (DOP-REF) for the conversion of physical to internal values. The referenced DATA-OBJECT-PROPs (including their UNITs) are also stored within the COMPARAM-SPEC.
COMPARAMs and COMPLEX-COMPARAMs have attributes to add semantic information necessary for the runtime system or application software. With the attribute PARAM-CLASS the communication parameter can be semantically classified in view of its usage. The PARAM-CLASS is mostly for documentational purposes only and does not carry a runtime semantics for the MCD 3D / MVCI Diagnostic Server. An exception are all communication parameters marked with the PARAM-CLASS UNIQUE_ID. These are used by the MCD 3D / MVCI diagnostic server to feed the D-PDU-API correctly with addresing information. The value set of PARAM-CLASS is extendable. In the following the corresponding predefined set of values is listed alongside with the values meanings:

— TIMING Message flow timing parameters, e.g. inter byte time or time between request and response.

— INIT Parameters for initiation of communication e.g. trigger address or wakeup pattern. These parameters must not be overwritten within ECU-Variant layer in any way.

— COM General communication parameter.

— ERRHDL Parameter defining the behaviour of the runtime system in case an error occurred, e.g. runtime system could either continue communication after a timeout was detected, or stop and reactivate.

— BUSTYPE This is used to define bustype specific parameters (e.g. baudrate). Most of these parameters affect the physical hardware. These parameters can only be modified by the first LOGICAL-LINK that acquired the physical resource. When a second LOGICAL-LINK is created for the same resource, these parameters that were previously set will be active for the new LOGICAL-LINK.

— UNIQUE_ID This type of communication parameter is used to indicate to both the ComLogicalLink and the Application that the information is used for protocol response handling from a physical or functional group of ECUs to uniquely define an ECU response.

An attribute named CPTYPE is used by the runtime system to take the right choice of behaviour if an unsupported comparam occurs. Unsupported means, that the runtime system has no knowledge about this communication parameter and how it has to be handled. Following values of CPTYPE are permitted:

— STANDARD The communication parameter specified by ISO 22900-2 for a standardized protocol (or defined by this document) has to be supported by the runtime system implementing the corresponding protocol. Diagnostic data using a standardized protocol not supported by the runtime system cannot be executed by the runtime system. If the runtime system detects an unsupported communication parameter of type STANDARD it has to stop the execution of the diagnostic data and to provide an error message.

— OEM-SPECIFIC The communication parameter is part of a non-standardized OEM-specific protocol; nevertheless it is required to be implemented by the runtime system. Diagnostic data using an OEM-specific protocol not supported by the runtime system cannot be executed by the runtime system. If the runtime system detects an unsupported communication parameter of type OEM-SPECIFIC it has to stop the execution of the diagnostic data and to provide an error message.

— OPTIONAL This communication parameter does not have to be supported by the runtime system. If a DIAG-COMM uses an unsupported comparam of this type the comparam can be ignored and the DIAG-COMM can be executed nevertheless.

The DISPLAY-LEVEL is used to restrict visibility (and therefore changeability) of the COMPARAMs in an application. Level 1 means that this can be changed for all users. Level 2 means that not everybody is allowed to change these communication parameter. The maximum value of DISPLAY-LEVEL is 5. The behaviour of the different applications according to the display levels can not be defined by the ODX standard. Therefore the support of DISPLAY-LEVEL is optional.
The attribute CPUSAGE has the possible values "ECU-SOFTWARE", "APPLICATION", "ECU-COMM" and “TESTER”. ECU-SOFTWARE parameters are communication parameters that are only used for ECU software generation and configuration. Communication parameters of this type shall be fully ignored by the MCD 3D / MVCI Diagnostic Server. APPLICATION parameters are only evaluated by the application. Communication parameters of this type shall not be passed down by the MCD 3D / MVCI Diagnostic Server to the PDU API, but may be used by the MCD 3D / MVCI Diagnostic server itself. In any case, they shall be accessible via the MCD 3D / MVCI Diagnostic

Server API. ECU-COMM parameters are parameters needed to communicate with the ECU (e.g. timings and addresses). A parameter of the kind “ECU-COMM” is relevant both to the tester during tester communication and to the ECU software generation / calibration use case. TESTER parameters are only valid to the tester during diagnostic communication, they are not relevant to the use case of ECU software generation and calibration.

7.3.4 INHERITANCE OF COMMUNICATION PARAMETERS

During runtime a diagnostic layer is linked with a PROT-STACK to import a valid set of communication parameters (COMPARAM and COMPLEX-COMPARAM elements). This unambiguous link is defined via the PROTOCOL or the LOGICAL-LINK. The COMPARAM values itself are passed on by value inheritance between the different layers. An inherited COMPARAM value is identical to that in the parent layer, i.e. it has the same value.
A communication parameter has to be overwritten to change its value. For that purpose the communication parameter is referenced by a COMPARAM-REF. The SIMPLE-VALUE or COMPLEX-VALUE of the COMPARAM-REF defines the new value(s) of the referenced COMPARAM or COMPLEX-COMPARAM. The PROTOCOL-SNREF can be used to define that the COMPARAM-REF should be applied only for the referenced protocol. If the PROTOCOL-SNREF is not set, the COMPARAM-REF is valid for all available protocols. The LINK-COMPARAM-REF at LOGICAL-LINK and PHYSICAL-VEHICLE-LINK is similar to COMPARAM-REF but without a PROTOCOL-SNREF. It is similar to the PROT-STACK-SNREF. The PROT-STACK-SNREF can be used to define that the COMPARAM-REF should be applied only for the referenced protocol stack. If the PROT-STACK-SNREF is not set, the COMPARAM-REF is valid for all available protocol stacks. A COMPARAM-REF detected at runtime that does not match the current PROTOCOL / PROT-STACK combination has no runtime relevance and can be ignored.

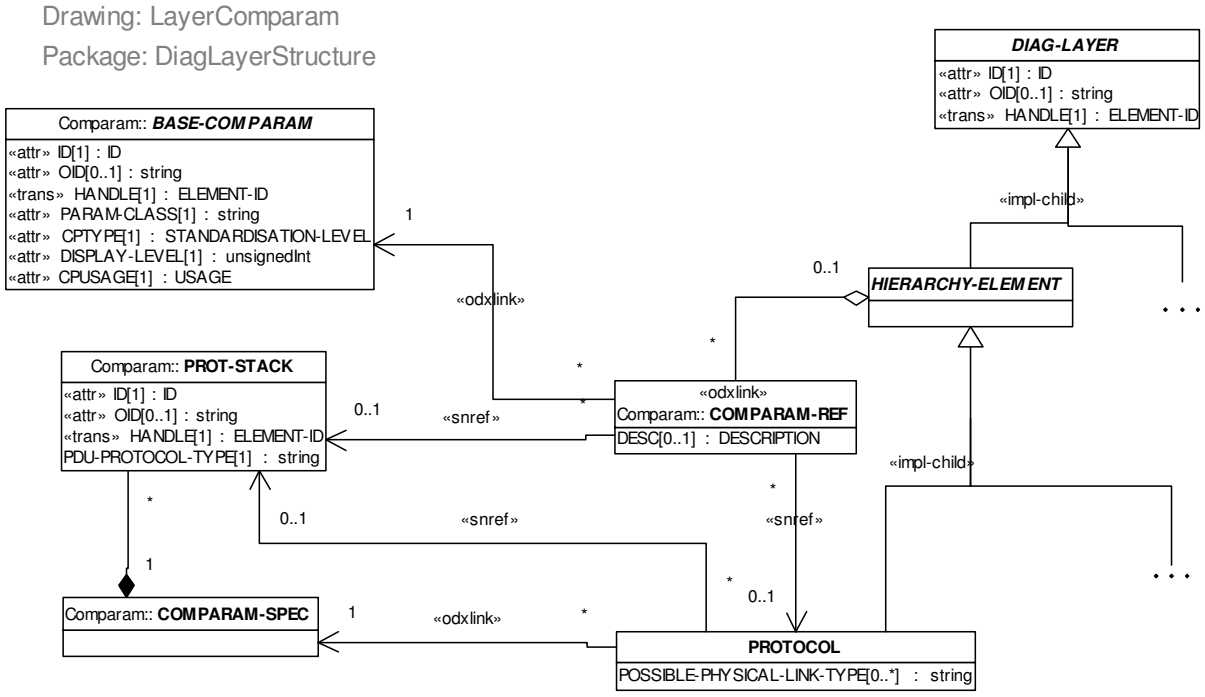


Figure 36 — Overwriting of communication parameters (incomplete structure)

If a communication parameter value is overwritten within the scope of a DIAG-LAYER then the changed communication parameter value is valid for all DIAG-COMMs that are available on this layer including DIAG-COMMs that were inherited from the parent layer. It is also possible to overwrite a COMPARAM value for a single DIAG-COMM. The scope of this change is restricted to this specific DIAG-COMM. communication parameter values for a specific DIAG-COMM are passed on implicitly by inheritance of the DIAG-COMM they belong to. Therefore, they can't be overwritten within the scope of a diagnostic layer. Only overwriting the whole DIAG-COMM can change them. If a communication parameter value is overwritten in both the DIAG-COMM and the diagnostic layer then the DIAG-COMMs DIAG-COMM-specific COMPARAM value takes precedence over the layer-specific one, i.e. the communication parameter value from the DIAG-COMM is used. Overwritten communication parameter values on a FUNCTIONAL-GROUP level are not inherited to lower layers (i.e. the inheriting base variants). Base variants take their set of communication parameter values directly from the protocol and then overwrite with locally defined values. Since a logical link always includes one dedicated protocol, all multiple inheritance issues can unambiguously be resolved at runtime.
An overview of the precedence rules for communication parameter values is shown in the table below (from lower to higher precedence):

Table 2 — Precedence rules for communication parameters

Location of value definition in ODX data model	Type of value definition
COMPARAM and COMPLEX-COMPARAM (in COMPARAM-SUBSET)	Default definition
diagnostic layer [PROTOCOL]	Overwrite
diagnostic layer [FUNCTIONAL-GROUP]	Overwrite (Attention: overwritten communication parameter values on FUNCTIONAL-GROUP level are not inherited to lower layers!)
diagnostic layer [BASE-VARIANT]	Overwrite (PROTOCOL layer)
diagnostic layer [ECU-VARIANT]	Overwrite (PROTOCOL layer, BASE-VARIANT layer)
PHYSICAL-VEHICLE-LINK	Overwrite
LOGICAL-LINK	Overwrite
DIAG-SERVICE	Overwrite

In case a communication parameter value is overwritten on the DIAG-SERVICE level, the runtime system has to make sure that, after a DIAG-SERVICE is completed, all communication parameters have the same values that they had before its execution.
EXAMPLE Simplified value inheritance example for simple communication parameters (COMPARAMs only)

As shown in the figure below, a PROTOCOL instance (PROT-A) references the PROT-STACK instance PS-B as the active PROT-STACK. As a consequence, the valid set of communication parameters consists of A, B and C. Because COMPARAM A value is overwritten within the protocol layer, the valid values in the scope of PROT-A are A=4, B=5 and C=15.

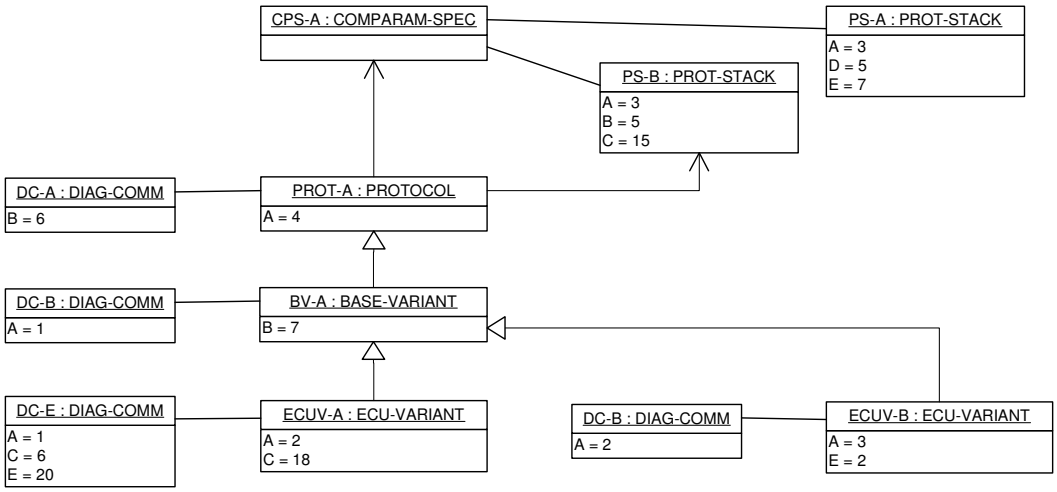


Figure 37 — Value inheritance of communication parameters (example)

These values apply to all the DIAG-COMM which don't overwrite a communication parameter must value themselves. The DIAG-COMM with the identifier DC-A in the PROTOCOL PROT-A shown in the example overwrites the COMPARAM B value. This is done within the data by a COMPARAM-REF element. The COMPARAM set for this special DIAG-COMM results in [A=4, B=6, C=15].
The BASE-VARIANT instance BV-A overwrites the COMPARAM B value. All other COMPARAM values are derived from the protocol layer PROT-A. The COMPARAM set in the scope of BV-A is [A=4, B=7 C=15]. Two DIAG-COMMs are available in this base variant layer. The DIAG-COMM DC-B defined within BV-A and the DIAG-COMM DC-A derived from the parent layer PROT-A. The COMPARAM set for DC-A at the base variant level is [A=4, B=6, C=15] and COMPARAM set for DC-B is [A=1, B=7, C=15].
There are two instances of an ECU-VARIANT both inherited from BV-A. For ECUV-A the COMPARAM set is [A=2, B=7, C=18]. There are three DIAG-COMMs available at ECUV-A. The value inherited DC-A with [A=2, B=6, C=18], the value inherited DC-B with [A=1, B=7, C=18] and locally defined DC-E with [A=1, B=7, C=6]. Because the COMPARAM E is not part of the current communication parameter set the COMPARAM-REF element at DC-E that tries to overwrite the COMPARAM E value exclusively for the DIAG-COMM has no affect and can be ignored. The ECU-VARIANT instance ECU-B applies the COMPARAM values [A=3, B=7, C=15]. E is ignored because it's not part of the active PROT-STACK. The available DIAG-COMMs are the value inherited DC-A with [A=3, B=6, C=15] and the locally defined DC-B which overwrites the DC-B defined at the layer above. The COMPARAM set for DC-B in the scope of ECUV-B is [A=2, B=7, C=15]. There is no other way around to change a COMPARAM value which is already overwritten at the DIAG-COMM in a parent layer then to overwrite the DIAG-COMM as a whole.

7.3.5 DATASTREAM

7.3.5.1 OVERVIEW

The Datastream chapter deals with the main topic of ODX. It specifies the diagnostic communication between a tester and an ECU. The communication process consists of two parts: request and response.
NOTE ODX only describes the PDU but not the header and checksum of the message. The service identifier is part of the PDU.

A request message is sent by the tester to the ECU e.g. if the tester wants to read or to write data. Normally, the ECU replies with a response message that contains the read data or the information about success of the writing operation. The following figure shows a simple communication between a tester and an ECU. The tester requests three bytes of data, which should be read starting at the memory address 204813h in the ECU's memory. The request message consists of three parameters: SID (service identifier), Address and Size and has the length of 5 bytes. The tester receives 4 bytes as the response to its request, which represent two parameters: SID and the requested three-byte value.

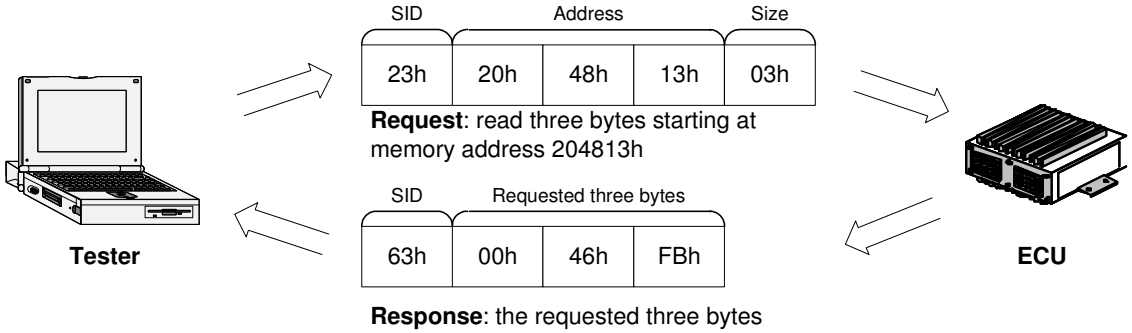


Figure 38 — Datastream example

The next figure shows the overview of the packages described in this chapter. Service, Response, Request and Parameters are already mentioned above. Another possibility to communicate with an ECU consists in using diagnostic jobs. They can be seen as complex services and, in fact, they are based on them. A job uses one or more services for communication and is useful for performing of more complex tasks than simple services can offer. Thus, Services and Jobs are two diagnostic communication primitives (DIAG-COMMs) used to communicate with an ECU. They are the two implementing peculiarities of the abstract class DIAG-COMM, which combines the common characteristics of jobs and services, described in the package DiagComm.

Drawing: Datastream
Package: Datastream

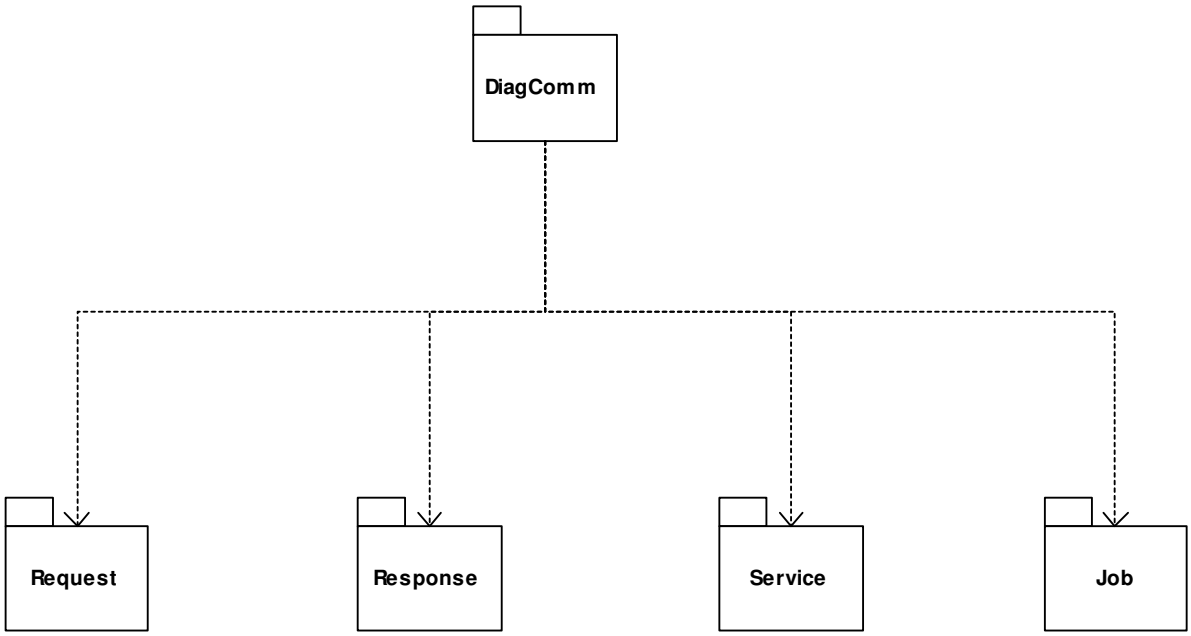


Figure 39 — Datastream

7.3.5.2 DIAGNOSTIC COMMUNICATION

A diagnostic communication (DIAG-COMM) provides common characteristics of diagnostic services (DIAG-SERVICEs) and jobs (SINGLE-ECU-JOBs). It defines a communication process between a tester and an ECU (or a group of ECUs) in order to enquire diagnostic information from the ECUs or/and to modify their behaviour for diagnostic purposes.

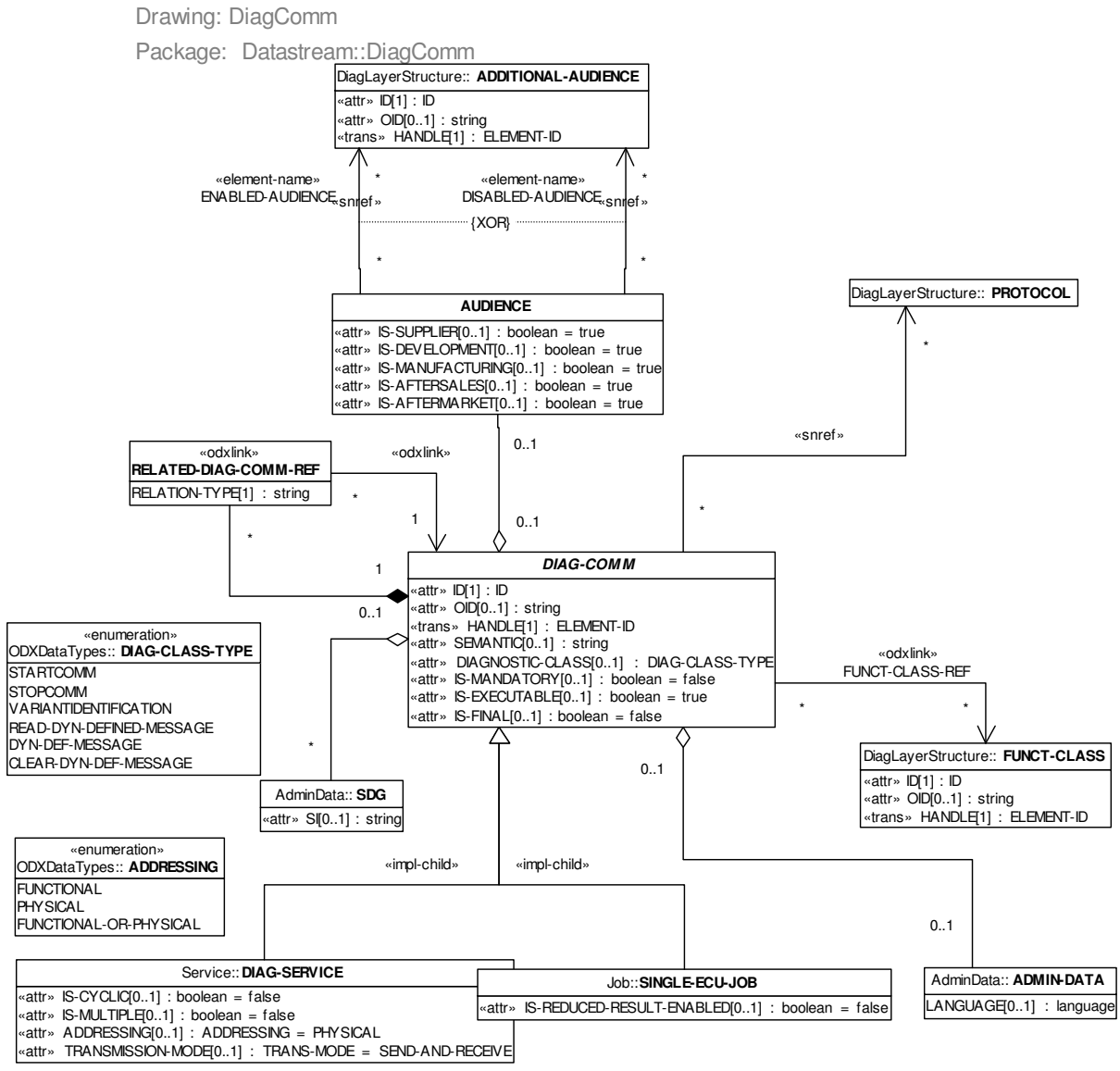


Figure 40 — Diagnostic communication

In addition to the standard members ID, HANDLE, ADMIN-DATA and SDGs, the following members are available for the characterisation of a DIAG-COMM (a service or a job):
AUDIENCE defines the target group or groups of users the DIAG-COMM is available for. For more details see 7.1.2.2.

IS-MANDATORY indicates whether the DIAG-COMM can be eliminated in lower diagnostic layers by using NOT-INHERITED-DIAG-COMM. The value "true" means that the DIAG-COMM must not be eliminated. By default each DIAG-COMM can be eliminated (IS-MANDATORY = "false").
IS-FINAL indicates whether the DIAG-COMM can be overridden in lower diagnostic layers by declaring a DIAG-COMM with the same SHORT-NAME. The value "true" means that the DIAG-COMM must not be overridden. By default each DIAG-COMM can be overridden (IS-FINAL = "false").
Some services or jobs can only be used as part of other jobs and it does not make sense to use them separately. IS-EXECUTABLE can be used in this case to tell the tool to hide this DIAG-COMM by setting IS-EXECUTABLE to "false". The default value of this attribute is "true".
Via RELATED-DIAG-COMM-REF it is possible to reference one or more DIAG-COMMs that are in relationship with this DIAG-COMM. It is merely used for information purposes. RELATION-TYPE indicates the user-specific type of the relation (e.g. INPUT, OUTPUT, SESSION or SECURITY).
The DIAGNOSTIC-CLASS attribute denotes the purpose of the DIAG-COMM and may be used by the run time system or other jobs for functional identification of a service or a job. E.g., to start the communication with an ECU a service with DIAGNOSTIC-CLASS="STARTCOMM" is used. Possible values for this attribute are: STARTCOMM, STOPCOMM, VARIANTIDENTIFICATION, DYN-DEF-MESSAGE, READ-DYN-DEF-MESSAGE and CLEAR-DYN-DEF-MESSAGE. STARTCOMM and STOPCOMM must not exist more than one time per layer. They do not have to be present, if the ECU does not need an explicit initialisation message.
The member SEMANTIC is used for classifying of services and jobs. The pre-defined values can be found in Table A.1 — SEMANTIC at DIAG-COMM.
A diagnostic service can redefine one ore more communication parameters by referencing an existing COMPARAM. A COMPARAM-REF contains a reference to a COMPARAM defined in a COMPARAM-SPEC and a new VALUE for this parameter. Additionally, there is a possibility to add a description to such a redefinition by means of DESC.
Each service or job can be assigned to a functional class using certain criteria. The assignment takes place by referencing the class with FUNCT-CLASS-REF. A service or a job may be assigned to one or more functional classes but also to no class. Such grouping of DIAG-COMM instances can be used by the application to sort them according to the defined criteria.
PROTOCOL-SNREF at DIAG-COMM is used to determine the protocol the DIAG-COMM is valid for. If no protocol is referenced all protocols in the transitive closure are supported. For DIAG-COMMs defined in an ECU-SHARED-DATA layer, it makes sense to define, under which protocol the DIAG-COMM can be executed. This information is given by referencing an arbitrary number of PROTOCOL instances via SHORT-NAME.

7.3.5.3 SERVICE

Diagnostic services (DIAG-SERVICEs) define a communication process between a tester and an ECU or a group of ECUs in order to enquire diagnostic information from the ECUs or/and to modify their behaviour for diagnostic purposes. The tester initialises a service by sending a request message. REQUEST-REF is used to reference an appropriate request object that describes the structure of the request message. If no error occurs, then the ECU replies by sending a positive response message. POS-RESPONSE-REF is used to reference the corresponding response object that describes the structure of the response message. In case of an error the tester receives one or more negative response messages. The description of them is done by NEG-RESPONSEs, which are referenced by NEG-RESPONSE-REFs from the DIAG-SERVICE.

A DIAG-SERVICE neither containing a POS-RESPONSE-REF nor a REQUEST-REF is not valid.

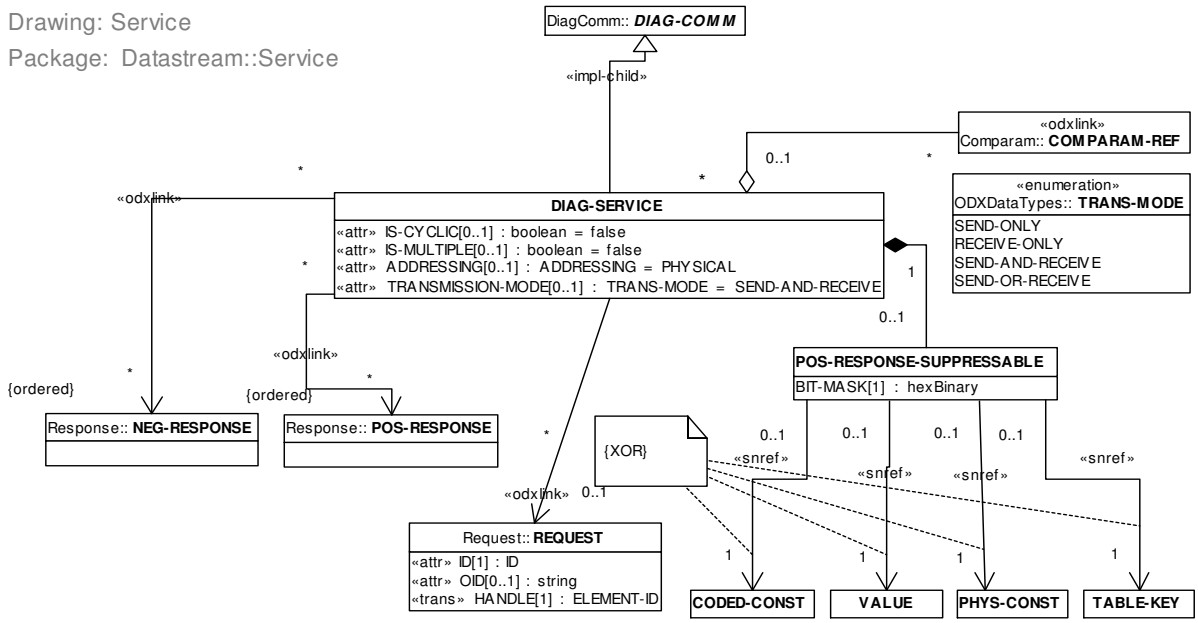


Figure 41 — Service

A DIAG-SERVICE inherits all the members defined within the DIAG-COMM class. Additionally, the members IS-CYCLIC and IS-MULTIPLE belong to a DIAG-SERVICE and it refers to related services, to a request and to positive and negative responses.
IS-CYCLIC indicates whether the service is used to request data that is sent repeatedly. For example a tester can request the machine speed that is to be sent at intervals of one second. The ECU replies on such a request by repeatedly sending responses to the tester where each response contains the actual machine speed. The default value of this member is "false" which indicates that no cyclic responses are to be expected.
Using some services multiple messages can be expected as responses to the request. In this case IS-MULTIPLE is to be set to "true". By default only a single response message is expected. As a stop criterion serves a timeout parameter. Use case for IS-MULTIPLE is handling protocols where e.g. DTCs are split up into multiple responses. In that case an MCDResult should contain multiple MCDResponse Objects in the MVCI Diagnostic Server.
Corresponding to the optional attribute TRANSMISSION-MODE the MCD 3D / MVCI Diagnostic Server will handle the Service in several ways. Possible values are:

— SEND-ONLY (The MCD 3D / MVCI Diagnostic Server sends a request message and will not expect any kind of response. Neither positive nor negative response. The classic use case is bus simulation.)

— RECEIVE-ONLY (The MCD 3D / MVCI Diagnostic Server will not send a request message. But shall listen for one of the referenced positive responses. The classic use case is for listening to onboard messages.)

— SEND-AND-RECEIVE (That is the regular diagnostic service. The MCD 3D / MVCI Diagnostic Server sends a request message and will listen for one of the referenced positive responses.)

— SEND-OR-RECEIVE (This value is a place holder. It can be changed to another transmission mode value during runtime before the DIAG-COMM is executed. Otherwise the DIAG-COMM will be executed in SEND-AND-RECEIVE transmission mode as default. Therefore DIAG-COMM of this type must be defined like a DIAG-COMM specified as SEND-AND-RECEIVE including request and response. The classic use case is for response-on-event handling.)

As mentioned above REQUEST-REF is used to reference the appropriate request while POS-RESPONSE-REFs reference the proper positive responses. NEG-RESPONSE-REFs reference negative responses that may be returned if an error occurs. It is possible to define more than one positive or negative response for one service. The MCD 3D / MVCI diagnostic server has to set up a table with expected responses that is passed down to the D-PDU-API. The information in this table is e.g. built from information retrieved from CODED-CONST and MATCHING-REQUEST paramaters. For details, please refer to the D-PDU API specification.
If inherited GLOBAL-NEG-RESPONSES and locally defined GLOBAL-NEG-RESPONSES are both valid in a specific location, the locally defined GLOBAL-NEG-RESPONSEs are evaluated before the inherited ones. In case of multiple inheritance the GLOBAL-NEG-RESPONSEs are evaluated in the alphabetical order of the SHORT-NAME of the DIAG-LAYER they have been inherited from.
If the POS-RESPONSE-SUPPRESSABLE element is defined at a DIAG-SERVICE it means the application can choose whether it expects a positive response or not. If the application wants no positive response the BIT-MASK of POS-RESPONSE-SUPPRESSABLE is applied with an OR-semantic to the CODED-VALUE of the parameter referenced by the POS-RESPONSE-SUPPRESSABLE element. If the element POS-RESPONSE-SUPPRESSABLE is not a sub-element of a DIAG-SERVICE it means the service is handled as usual.
EXAMPLE Diagnostic service with a POS-RESPONSE-SUPPRESABLE element. In case the positive response should be suppressed the following would happen: The BIT-MASK 80hex is applied with an OR operation to the sub-function 18 (12hex). As the result 92hex is sent instead of 12hex as subfunction.

```
<DIAG-COMMS>
	<DIAG-SERVICE ID="ExampleServiceID" ADDRESSING="FUNCTIONAL-OR-PHYSICAL">
		<SHORT-NAME>ExampleService</SHORT-NAME>
		<AUDIENCE/>
		<REQUEST-REF ID-REF="ExampleRequestID"/>
		<POS-RESPONSE-REFS>
			<POS-RESPONSE-REF ID-REF="ExampleResponseID"/>
		</POS-RESPONSE-REFS>
		<POS-RESPONSE-SUPPRESSABLE>
			<BIT-MASK>80</BIT-MASK>
			<CODED-CONST-SNREF SHORT-NAME="ParamOfSubFunction"/>
		</POS-RESPONSE-SUPPRESSABLE>
	</DIAG-SERVICE>
</DIAG-COMMS>
<REQUESTS>
	<REQUEST ID="ExampleRequestID">
		<SHORT-NAME>ExampleRequest</SHORT-NAME>
		<PARAMS>
			<PARAM xsi:type="CODED-CONST">
				<SHORT-NAME>SIDParam</SHORT-NAME>
				<CODED-VALUE>15</CODED-VALUE>
				<DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"/>
			</PARAM>
			<PARAM xsi:type="CODED-CONST">
				<SHORT-NAME>ParamOfSubFunction</SHORT-NAME>
```

```
                        <CODED-VALUE>18</CODED-VALUE>
                        <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"/>
                    </PARAM>
                </PARAMS>
            </REQUEST>
</REQUESTS>
```

ADDRESSING is used to define the addressing mode used by the DIAG-SERVICE. The valid values are FUNCTIONAL, PHYSICAL or FUNCTIONAL-OR-PHYSICAL whereby the default value is PHYSICAL. FUNCTIONAL means the DIAG-COMM will be executed using a functional address. In the case of PHYSICAL only an individual ECU can be addressed using its physical address. Finally, FUNCTIONAL-OR-PHYSICAL is used when both modes are supported.
The functional address is only be used in the Level of the FUNCTIONAL-GROUP layer. When communicating functional, it is possible to open a logical link to a functional group. Now, the functionally executed service is used as defined at this FUNCTIONAL-GROUP. However, the the service could be overridden on BASE-VARIANT level. The meaning of the DIAG-SERVICE overriding with respect to the functional addressing is described in the table below.

Table 3 — Overriding of functional services

FUNCTIONAL-GROUP ADDRESS	BASE-VARIANT ADDRESS	Semantics
FUNCTIONAL	FUNCTIONAL	Response message is only overridden for the functional case. The service shall not be offered as an executable service on the BASE-VARIANT or ECU-VARIANT level. An overridden request message is ignored.
FUNCTIONAL	FUNCTIONAL-OR-PHYSICAL	Response message overrides for functional and physical addressing. Request only overrides for physical addressing. On BASE-VARIANT or ECU-VARIANT level the service can only be executed physically.
FUNCTIONAL	PHYSICAL	Response message overrides for physical addressing only. For functional addressing, the response of the functional group is used. Request only overrides for physical addressing. On BASE-VARIANT or ECU-VARIANT level the service can only be executed physically.
FUNCTIONAL-OR-PHYSICAL	FUNCTIONAL-OR-PHYSICAL	Response message overrides for functional and physical addressing. Request only overrides for physical addressing. On BASE-VARIANT or ECU-VARIANT level the service can only be executed physically.
FUNCTIONAL-OR-PHYSICAL	PHYSICAL	Response message overrides for physical addressing only. For functional addressing, the response of the functional group is used. Request only overrides for physical addressing. On BASE-VARIANT or ECU-VARIANT level the service can only be executed physically.

NOTE Usage of Base-Variant overridden response messages can only be used after base variants have been identified

7.3.5.4 PARAMETER

A response or a request consists of one or more parameters (PARAMs). There are several types of parameters:

— VALUE

— RESERVED

- CODED-CONST
- PHYS-CONST
- LENGTH-KEY
- MATCHING-REQUEST-PARAM
- TABLE-KEY
- TABLE-STRUCT
- TABLE-ENTRY
- DYNAMIC
- SYSTEM
- NRC-CONST

All of them can be used directly or indirectly in a response and in a request except that MATCHING-REQUEST-PARAM and DYNAMIC may only be used in a response.
Matching of result parameters can be done in one of the following ways:

- MATCHING-REQUEST-PARAM (repetition of data from request)
- CODED-CONST (matching with fixed coded-value)
- PHYSICAL-CONST (matching coded value has to be calculated with DOP)

awing: Parameter
ckage: Datastream::Service

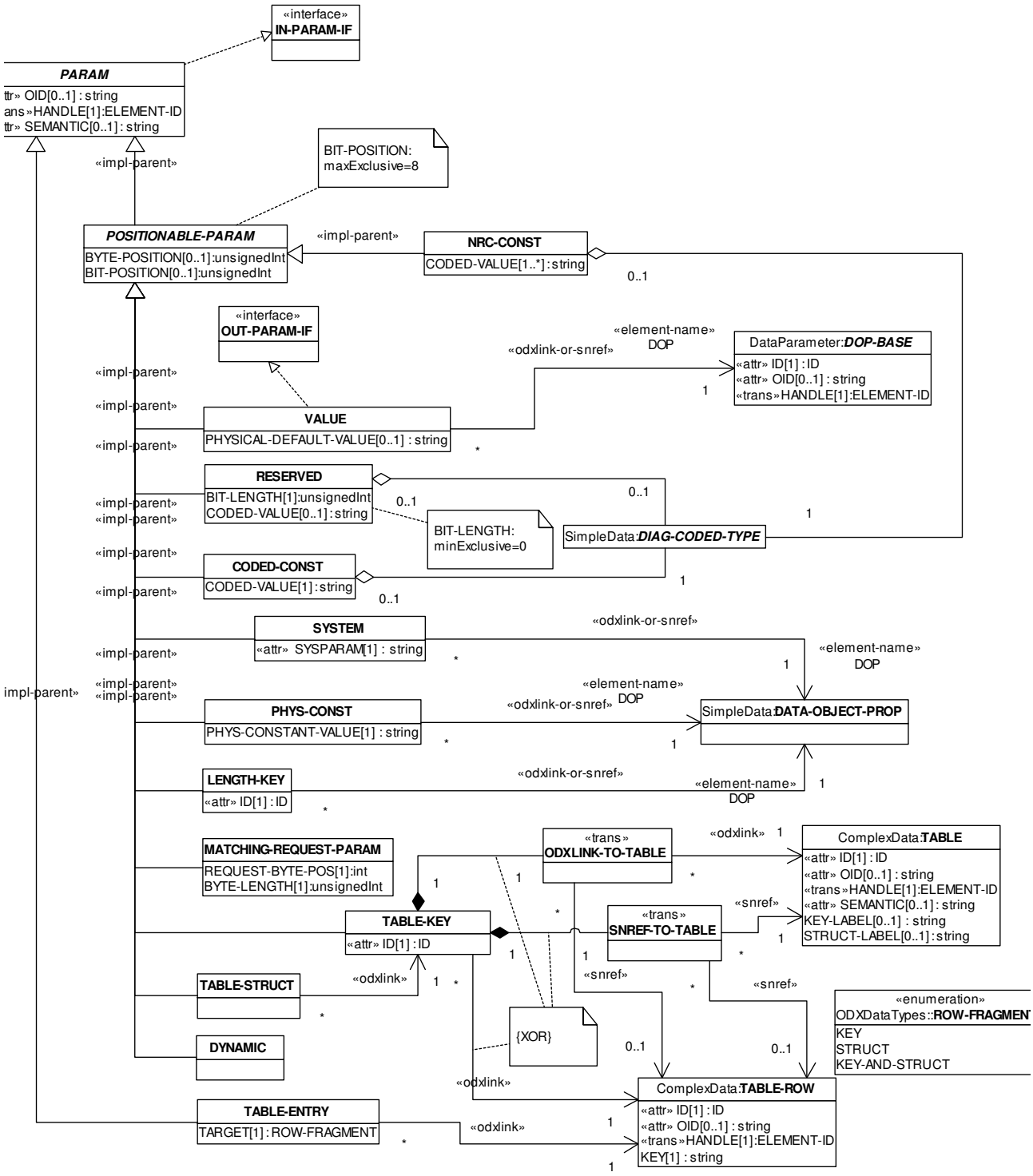


Figure 42 — Parameter

A VALUE parameter is a frequently used parameter type. It references a DOP, which converts a concrete value from coded into physical representation and vice versa. The

member PHYSICAL-DEFAULT-VALUE can be used by this parameter to define the default value to be sent to the ECU in case of a request, if the user / the application specifies no value. In a response message PHYSICAL-DEFAULT-VALUE is used for verification of the received value. In this case, the received coded value must be converted into the physical representation. After that the specified and the received values can be compared.
A parameter of type RESERVED is used when the parameter should be ignored by the runtime system. Such parameters are not interpreted and are not shown on the user's display. If the last or the first parameter of the PDU (request or response) should not be interpreted, it must not be omitted. It must be defined in the ODX instance as RESERVED in order to compute the PDU length and to compare it with the length of the received PDU in case of a response. If RESERVED is used in a request with CODED-VALUE and DIAG-CODED-TYPE the parameter should be processed like a CODED-CONST parameter. If CODED-VALUE and DIAG-CODED-TYPE are not defined, the tester has to fill the parameter with an arbitrary value. An MVCI server shall write a 0 into every bit in a request covered by a RESERVED parameter.
CODED-CONST parameters are used in a request, if the user must not change its value. For example, the service identifier is always fixed. In this case DIAG-CODED-TYPE provides information how to put the given value into the PDU. In a response it can be used for verification of the received value without converting it into the physical representation.
PHYS-CONST is an alternative for CODED-CONST with the difference that the value is given in the physical representation. In a request it must be converted into the coded value using the referenced DATA-OBJECT-PROP. In the response, either the given physical value must be converted into the coded representation or the received coded value must be converted into the physical representation. After that the specified and the received values can be compared.
In some cases the parameter's length can be defined by another one in the same message. For this purpose the type LENGTH-KEY exists. Such a parameter has the member ID and is referenced by the DIAG-CODED-TYPE of the DATA-OBJECT-PROP used by the parameter, whose length is determined by the LENGTH-KEY parameter. The LENGTH-KEY parameter and the parameter using the DATA-OBJECT-PROP that refers to this LENGTH-KEY parameter should be defined within the same wrapper (STRUCTURE, ENV-DATA, RESPONSE or REQUEST). The section 7.4.8 gives an example for use of LENGTH-KEY parameters.
The verification can also take place by matching the received value with a value in the request message using MATCHING-REQUEST-PARAM. REQUEST-BYTE-POS and BYTE-LENGTH are used then to define the position and the length of the value within the request PDU. The counting of REQUEST-BYTE-POS starts with zero at the first byte of the PDU. In this case the request PDU must be stored by the runtime system for the comparison. Furthermore, the received value can be compared only with byte-aligned values within the request PDU because no bit wise value extraction takes place from the request PDU. MATCHING-REQUEST-PARAM may only be used in a response.
The parameters of types TABLE-KEY, TABLE-STRUCT and TABLE-ENTRY are used for reading and writing data by data identifiers. Parameter type TABLE-ENTRY can only exist at a STRUCTURE referenced by TABLE-ROW. TABLE-KEY and TABLE-STRUCT parameters should only be used within the same PARAMS wrapper (of a STRUCTURE, an ENV-DATA, a RESPONSE or a REQUEST). For more information about them see section 7.3.6.11.
With some protocols (e.g. ISO 14230 or ISO 14229-1) it is possible to dynamically define data records at run time, which contain values from other records identified by a local identifier, a common identifier or an address in the ECU memory, etc. A DYNAMIC parameter is used in this case to indicate the runtime system that the parameter was defined dynamically. If the runtime system detects a DYNAMIC parameter it has to

interpret the parameter according to the information stored in the memory. See section 7.4.2 to get an example for use of DYNAMIC parameters.
The parameter of type SYSTEM is used to cause the runtime system to calculate a value that depends on the run time information (e.g. the current system time) or on the special runtime system information (e.g. the tester ID). The member SYSPARAM defines this value type. All SYSPARAMS listed below must be supported by any runtime system. This value set of SYSPARAM is extendable. If a user-defined SYSPARAM is used, which is not supported by the runtime system, an error has to be reported. The value received by the runtime system is a physical value and must be coded using the referenced DOP. Within a response the handling of parameters of type SYSTEM is similar to the handling of parameters of type VALUE.
The following data types and codings apply for SYSPARAMs:

Table 4 — Coding of system parameter

SYSPARAM	data type (physical)	coding	example
TIMEZONE	A_INT32	Count of minutes to UTC	+60 (Berlin)
YEAR	A_UINT32	YYYY	2004
MONTH	A_UINT32	MM	03 (march)
DAY	A_UINT32	DD	01 (first day of month)
HOUR	A_UINT32	hh	22 (10 pm)
MINUTE	A_UINT32	mm	00 (full hour)
SECOND	A_UINT32	ss	00 (full minute)
TESTERID	A_BYTEFIELD		"00F056"
USERID	A_BYTEFIELD		"043FF0"
CENTURY	A_UINT32	CC	20 (in year 2004)
WEEK	A_UINT32		04 (Fourth week of the year)
TIMESTAMP	A_BYTEFIELD	The number of elapsed milliseconds since 1970-01-01 00:00:00 GMT coded as 64bit-Integer (most significant byte first)	"000001000FFFFFFF" for the date 2004-11-06 22:27:43 GMT

BYTE- and BIT-POSITION specify the start position of the parameter in the PDU, where the counting starts with zero. This information is used for assembling of the request message and for extraction of the parameter from the response PDU. Sometimes, the position of a parameter cannot be determined until runtime, so the BYTE- and the BIT-POSITION must be omitted in the ODX instance. The parameter starts then at the byte edge next to the end of the last parameter. The MVCI server has to pack all PARAMs without a BYTE-POSITION in the exact order in which they are specified within the ODX file. The same applies to data extraction: All unpositioned parameters are extracted in the order in which they are specified in ODX. An unpositioned parameter is always positioned after the last parameter that has been processed by the MVCI server.
For more information about data extraction see section 7.3.6.2. BIT-POSITION must not have a value greater than 7.
Information about the content of a parameter is defined by the member SEMANTIC. The values are listed in Table A.2 — SEMANTIC at PARAM.
All parameters of statically undetermined length (e.g. parameters with MIN-MAX-LENGTH type, LEADING-LENGTH-INFO or of PARAM-LENGTH-INFO-TYPE) need to be byte-aligned. In other words a set of 1 to many bytes is covered. For all parameters with COMPLEX-DOPs associated the same constraint applies.
The list can also be extended if needed. This member can be used by the application to handle the parameter in a specific way or for searching and sorting purposes.

7.3.5.5 REQUEST

The request message is an integral part of each diagnostic service. It is sent by the tester to an ECU or a group of ECUs in order to get some diagnostic data or to adjust the ECU's settings. A request describes the structure of such a request message. It consists of one or more request parameters (PARAMs) that define the content of the message. The ECU interprets these parameters and replies to the tester by sending a response message, which consists of response parameters in an analogous manner.

Drawing: Request

Package: Datastream::Request

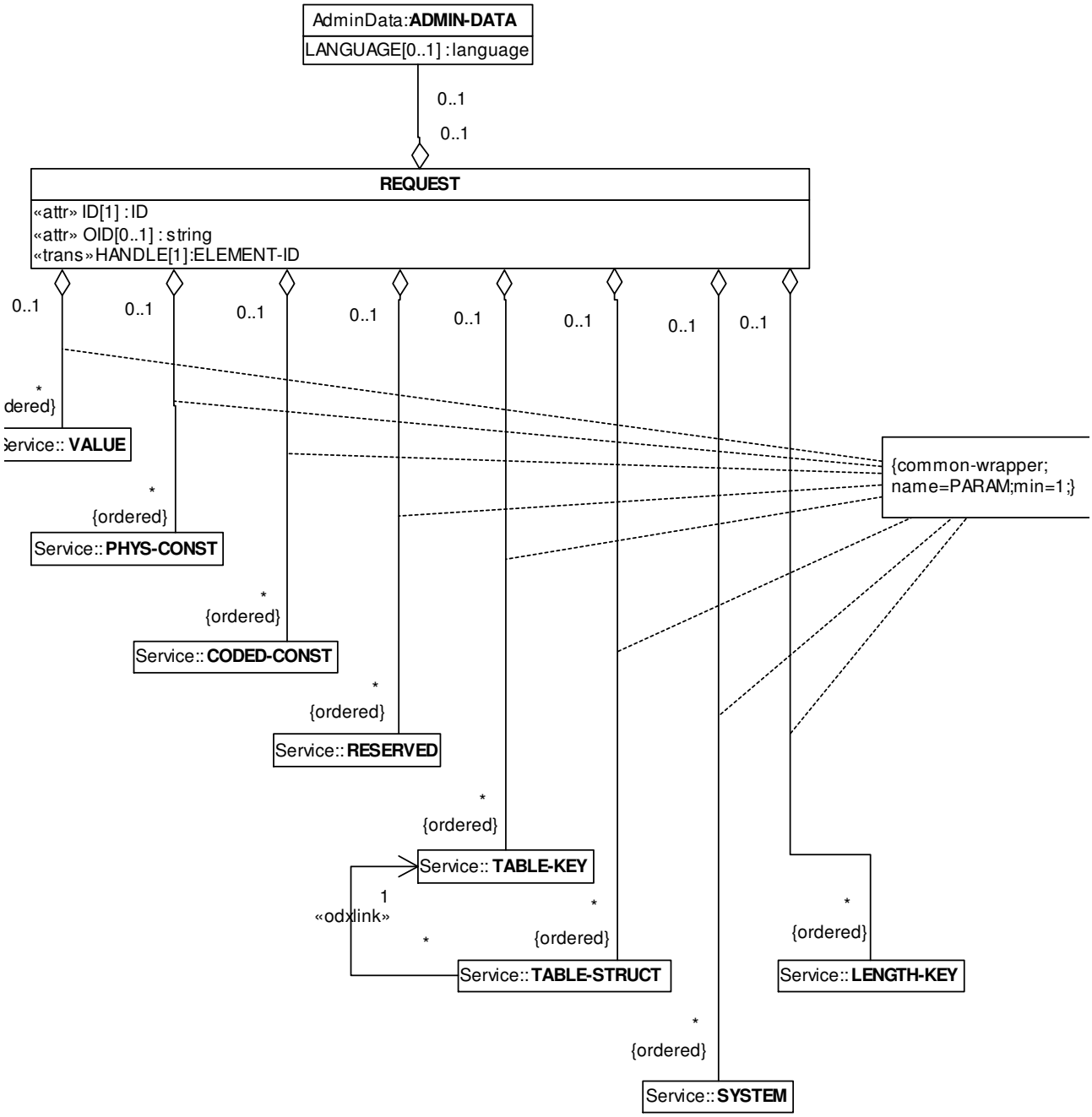


Figure 43 — Request

Each request owns the member ID that unambiguously identifies the request. One or more PARAM objects give the content of the request message. The PARAMs must not overlap in the request byte stream and all the bytes and bits of the request byte stream must be covered by one of the request parameters.

Generally, the request message is assembled as follows. Firstly, each parameter is converted to its coded representation if needed. Then the coded values are put into the PDU at the specified BYTE- and BIT-POSITION. Bytes are counted from zero starting with the first byte of the message, i.e. byte 0 is the first byte after the message header. Bits are counted from zero starting with the least significant bit (LSB). If the BIT-POSITION is omitted, the coding starts at the bit position 0 of the specified byte.
EXAMPLE ReadDataByAddress request message defined in Keyword Protocol 2000:

```
<REQUEST ID="REQ_ReadDataByAddress">
       <SHORT-NAME>REQ_ReadDataByAddress</SHORT-NAME>
       <LONG-NAME>Read Data By Address Request</LONG-NAME>
       <PARAMS>
              <PARAM SEMANTIC="SERVICE-ID" xsi:type="CODED-CONST">
                     <SHORT-NAME>ServiceId</SHORT-NAME>
                     <BYTE-POSITION>0</BYTE-POSITION>
                     <CODED-VALUE>35</CODED-VALUE>
                     <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"
xsi:type="STANDARD-LENGTH-TYPE">
                            <BIT-LENGTH>8</BIT-LENGTH>
                     </DIAG-CODED-TYPE>
              </PARAM>
              <PARAM xsi:type="VALUE">
                     <SHORT-NAME>Address</SHORT-NAME>
                     <LONG-NAME>Address in the ECU memory</LONG-NAME>
                     <BYTE-POSITION>1</BYTE-POSITION>
                     <BIT-POSITION>0</BIT-POSITION>
                     <DOP-REF ID-REF="DOP_3ByteIdentical"/>
              </PARAM>
              <PARAM xsi:type="VALUE">
                     <SHORT-NAME>Size</SHORT-NAME>
                     <LONG-NAME>Size of the value</LONG-NAME>
                     <BYTE-POSITION>4</BYTE-POSITION>
                     <DOP-REF ID-REF="DOP_1ByteIdentical"/>
              </PARAM>
       </PARAMS>
</REQUEST>
```

The first PARAM (the service ID) has a constant internal value and cannot be modified by the user. The user / application must specify the second and the third PARAMs because neither a constant nor a default value is given. The second parameter ("Address") references the DOP with ID = "DOP_3ByteIdentical", while the third one ("Size") references the DOP with ID="DOP_1ByteIdentical". Both DOPs convert the values given by the user / application into their coded representation.

7.3.5.6 RESPONSE

A response message is the ECU's part of a diagnostic service. It is returned by the ECU in answer to a tester's request message. The class RESPONSE describes the structure of such a response message. Three types of responses are defined in ODX: positive, negative and global negative responses. Normally a positive response message (POS-RESPONSE) is sent by an ECU. If an exceptional situation occurs within the ECU, a negative response message can be sent back.
Negative responses might be defined specific to a service or to a diagnostic layer instance. In the first case the negative response shall be defined by referencing a NEG-RESPONSE from the service via NEG-RESPONSE-REF. In the second case, a global negative response (GLOBAL-NEG-RESPONSE) shall be defined. This response shall automatically be used by a runtime system if no negative response, referenced by the actual service, matches.

Drawing: Response
Package: Datastream::Response

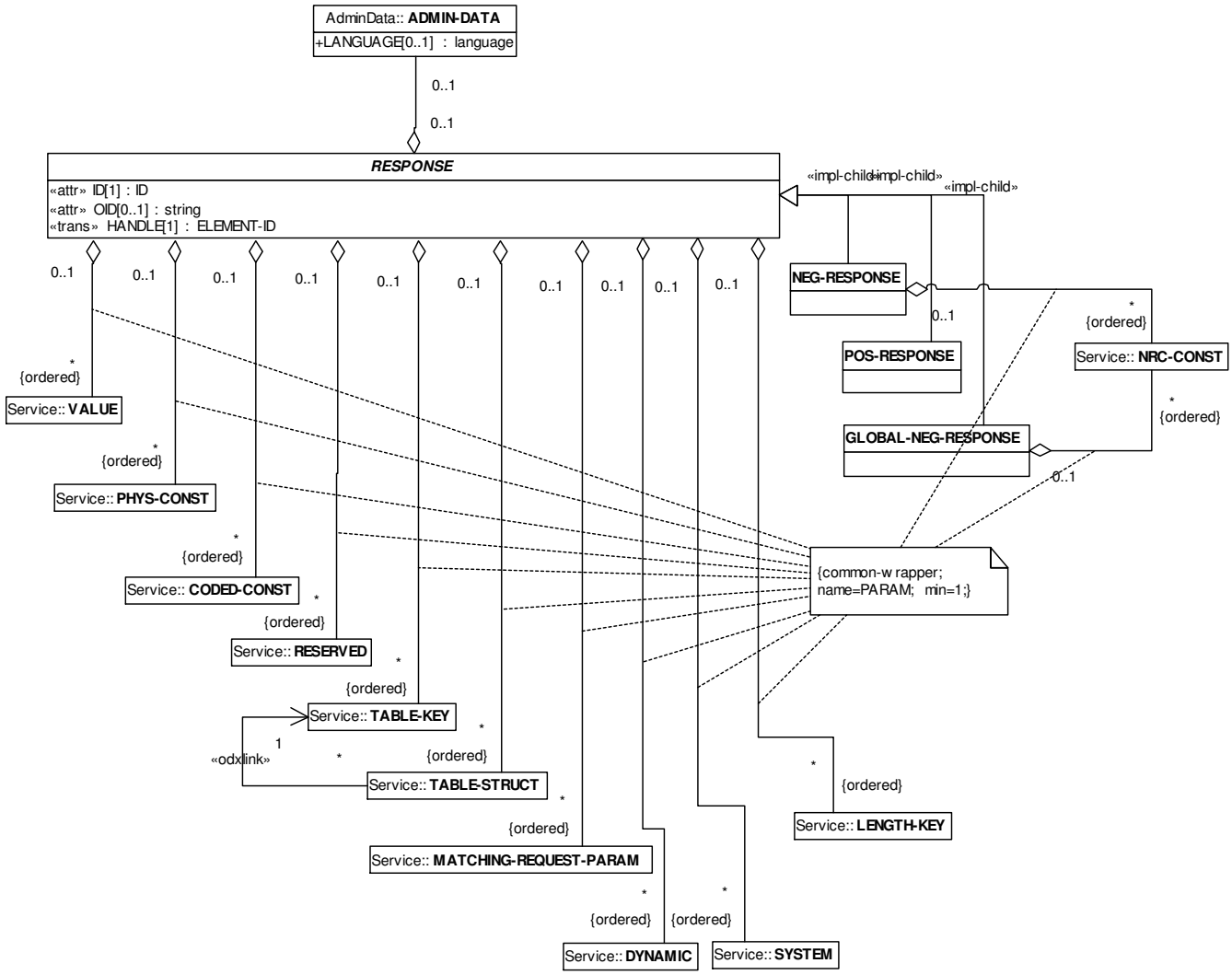


Figure 44 — Response

All types of responses have the same structure. A response is identified by the ID attribute and consists of several response parameters (PARAMs) that represent all data to be extracted from the response message.
The interpretation of the response consists of the interpretation of the parameters defined by the RESPONSE object. See section 7.3.5.4 for more information about the parameters.

EXAMPLE ReadDataByAddress response message defined in Keyword Protocol 2000:

```
<POS-RESPONSE ID="RESP_ReadDataByAddress">
       <SHORT-NAME>RESP_ReadDataByAddress</SHORT-NAME>
       <LONG-NAME>Read Data By Address Response</LONG-NAME>
       <PARAMS>
              <PARAM SEMANTIC="SERVICE-ID" xsi:type="CODED-CONST">
                     <SHORT-NAME>RDBA</SHORT-NAME>
                     <LONG-NAME>read Data By Address </LONG-NAME>
                     <BYTE-POSITION>0</BYTE-POSITION>
                     <CODED-VALUE>99</CODED-VALUE>
                     <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"
xsi:type="STANDARD-LENGTH-TYPE">
                            <BIT-LENGTH>8</BIT-LENGTH>
                     </DIAG-CODED-TYPE>
              </PARAM>
              <PARAM SEMANTIC="DATA" xsi:type="VALUE">
                     <SHORT-NAME>ReadData</SHORT-NAME>
                     <LONG-NAME>Read data</LONG-NAME>
                     <BYTE-POSITION>1</BYTE-POSITION>
                     <DOP-REF ID-REF="DOP-ByteArray"/>
              </PARAM>
       </PARAMS>
</POS-RESPONSE>
```

In this example the first PARAM is the service identifier. Its value must match the value given by CODED-VALUE and DIAG-CODED-TYPE. The second PARAM references a DOP with ID = "DOP-ByteArray", which interprets the received byte array.

7.3.5.7 JOB

Jobs (SINGLE-ECU-JOBs) are a second kind of diagnostic communication primitive (DIAG-COMM) next to services (DIAG-SERVICE, see above). They are seen as complex services that are not provided natively by the ECU, but have to be implemented manually on top of the ECU's native services. Jobs solely interact with the MCD 3D / MVCI Diagnostic Server API. They have no means to communicate directly with e.g. a diagnostic application or a user interface for interaction. Thus, they are macros for recurring complex tasks (e.g. reading of diagnostic trouble codes and the corresponding environment data, seed&key algorithms or ECU memory programming). Jobs are not to be used to define complete diagnostic test sequences.
The MCD 3D / MVCI Diagnostic Server API supports two kinds of jobs, which are modelled separately within ODX. One kind of job contains calls to services of one and only one dedicated ECU (SINGLE-ECU-JOB). These jobs are explained in this chapter. The service calls within the other kind (MULTIPLE-ECU-JOB) span multiple ECUs and are explained in a separate chapter, because they cannot be assigned to one DIAG-LAYER alone.

Drawing: Job
Package: Datastream::Job

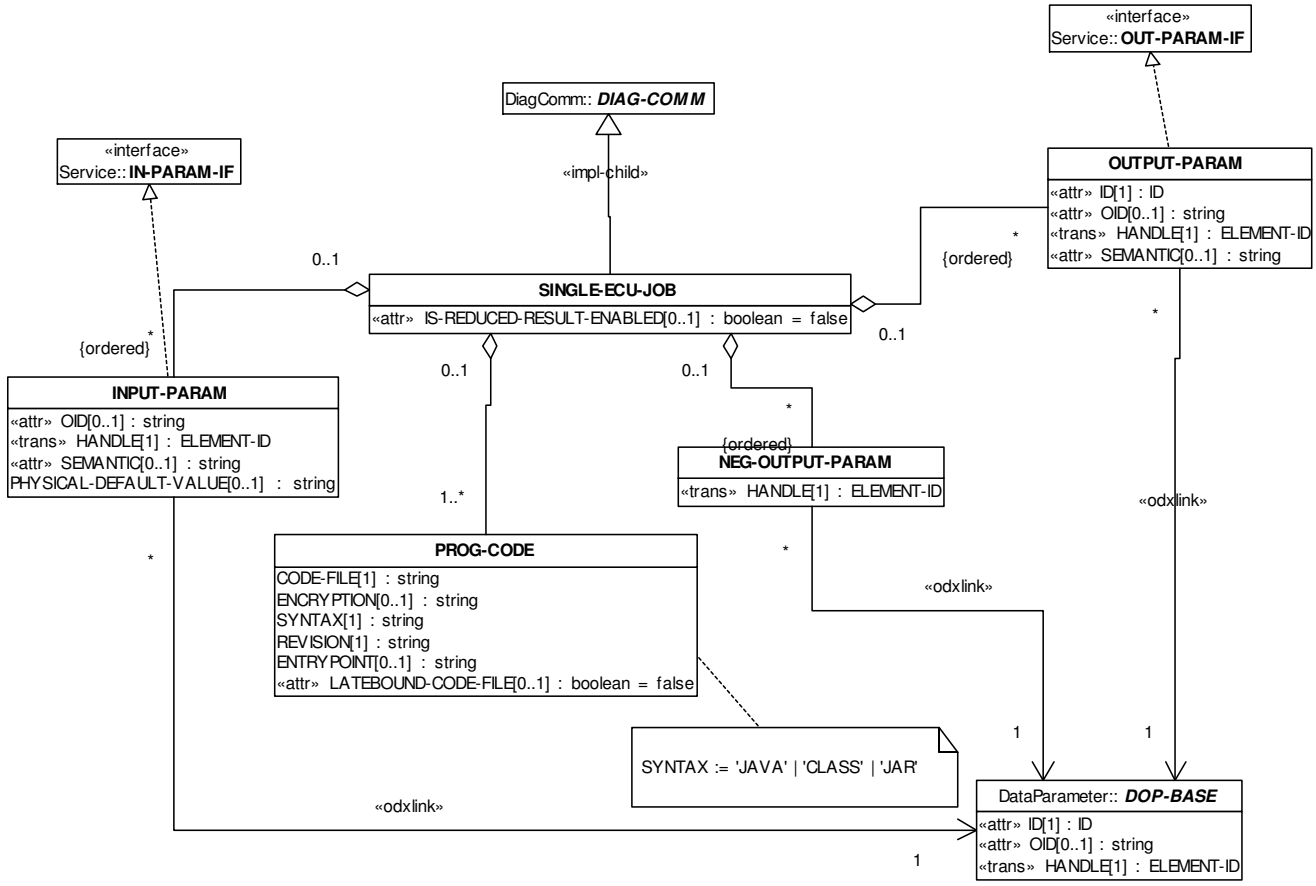


Figure 45 — Job

A SINGLE-ECU-JOB is derived from DIAG-COMM, the generic class for all diagnostic communication primitives. In analogy to a DIAG-SERVICE, a SINGLE-ECU-JOB has input and output data, which are modelled as a set of input parameters (INPUT-PARAM) and output parameters (OUTPUT-PARAM). In case of errors, a set of negative output parameters (NEG-OUTPUT-PARAM) may be returned by the job instead of a set of regular output parameters.
Job INPUT- and OUTPUT PARAMs have references to a DOP-BASE. However DOPs are applied differently from services: The conversion method is ignored, meaning an input parameter is passed to the job as-is and always as a physical value. If both, PHYSICAL-DEFAULT-VALUE and PHYSICAL-CONSTANT-VALUE are used within the same INPUT-PARAM instance the PHYSICAL-DEFAULT-VALUE carries no semantic and is to be ignored by an interpreting tool (e.g. the MCD 3D / MVCI diagnostic server). PHYSICAL-DEFAULT-VALUE and PHYSICAL-CONSTANT-VALUE are both ignored, if INPUT-PARAM references a COMPLEX-DOP. Output parameters are returned by the job as physical values. The physical values returned by the job must comply with the PHYSICAL-TYPE specification of the associated DOP-BASE.

Special cases are input parameters that have a TEXTTABLE defined as COMPU-METHOD. Here, the valid texts of the TEXTTABLE can be listed within an application. Upon selection, the corresponding text (the physical value) is passed to the job. Hence, the COMPU-METHOD is only used for comfort reasons not for actual conversion of the input parameter from a physical into a coded type. By consequence, the DIAG-CODED-TYPE of the DOP associated by a job parameter (regardless of whether it is an INPUT- or OUTPUT-PARAM) is ignored. The Reasons for this solution are as follows:

- The job parameter specification can reuse DOPs of contained services in cases where the service parameters are part of the job's input or output as well.
- The MCD 3D / MVCI Diagnostic Server does not support conversion of input- and output parameters of jobs.

The SEMANTIC attribute's value domain of the INPUT- and OUTPUT-PARAM is defined in analogy to that of PARAMs.
In the case of IS-REDUCED-RESULT-ENABLED is "true" the MCD 3D / MVCI Diagnostic Server is allowed to deliver a reduced parameter set within the response.
The core aspect of a job is the program code (PROG-CODE). A PROG-CODE instance contains all data needed to reference a specific piece of code that is to be executed as the job. Within the PROG-CODE instance, a reference to the file containing the code is specified (CODE-FILE). The syntax of the code is given in SYNTAX (either JAVA for Java source code, or CLASS for Java byte code or JAR for .jar-files) and the code revision number in REVISION. Optionally, an encryption algorithm may be specified in ENCRYPTION as well as an ENTRYPOINT into a binary code or .jar-file.
For the naming of a java job used in PROG-CODE there are some guidelines to be observed. To assure the unambiguousness of the name of the used java classes the namespace of java package will be used. The package structures are supplier specific and may start for example with company's internet domain. The following regulations exist for the different SYNTAX values:

- In case of SYNTAX = "JAVA" the CODE-FILE value must be specified using the slash character as package separator. For example "com/carmanufacturer/jobs/EcuJob.java".
- In case of SYNTAX = "CLASS" the CODE-FILE must be specified using the dot character as package separator. For example "com. carmanufacturer.jobs.EcuJob".
- In case of SYNTAX = "JAR" the CODE-FILE must contain the name of the Jar-file and the ENTRY-POINT specifies the name of the job in dot notation. Example: CODE-FILE = "jobs.jar" and ENTRY-POINT = "com.carmanufacturer.jobs.EcuJob".

However, if jobs in another syntax (e.g. a Windows DLL) need to be supported, a corresponding value for SYNTAX and ENTRY-POINT can be used. A regular MCD 3D / MVCI Diagnostic Server implementation is not able to execute such a proprietary job. Similar information applies to the ENCRYPTION field.
EXAMPLE 1: In the following, an example for a SINGLE-ECU-JOB instance is given. The data is meant for a job able to return a list of all diagnostic trouble codes for a KW2000 ECU. An input parameter allows to optionally retrieve the diagnostic trouble codes together with the environment data. Thus, the job contains calls to two KW2000 services of the ECU. The output parameter has a reference to a DOP. In this case, it is a COMPLEX-DOP describing the data structure for the returned data of the ECU's fault memory.

```
<SINGLE-ECU-JOB ID="JOB_Read_DTC_With_Env_Data" SEMANTIC="FAULTREAD">
       <SHORT-NAME>JOB_Read_DTC_With_Env_Data</SHORT-NAME>
       <AUDIENCE/>
       <PROG-CODES>
              <PROG-CODE>
                     <CODE-FILE>SEJ_ReadDTCWithEnvData.java</CODE-FILE>
                     <SYNTAX>JAVA</SYNTAX>
                     <REVISION>1.1.2</REVISION>
```

```
            </PROG-CODE>
        </PROG-CODES>
        <INPUT-PARAMS>
            <INPUT-PARAM>
                <SHORT-NAME>WITH_ENV_DATA</SHORT-NAME>
                <PHYSICAL-DEFAULT-VALUE>true</PHYSICAL-DEFAULT-VALUE>
                <DOP-BASE-REF ID-REF="ID_73474"/>
            </INPUT-PARAM>
        </INPUT-PARAMS>
        <OUTPUT-PARAMS>
            <OUTPUT-PARAM ID="ID_73472">
                <SHORT-NAME>DTC_LIST</SHORT-NAME>
                <DOP-BASE-REF ID-REF="ID_73473"/>
            </OUTPUT-PARAM>
        </OUTPUT-PARAMS>
</SINGLE-ECU-JOB>
```

EXAMPLE 2: In a second example the data of a job is shown that calls a single diagnostic service. This service is the ReadDataByAddress service already used as an example in section 7.3.5.3 ff. (there it is named ReadMemoryByAddress). As an essential difference between the job's parameters and the parameters of the service's request that is called in its body, constant parameters do not have to be included. Constant parameters of the service will be filled with their constant value automatically when the job calls the service via the MCD 3D / MVCI Diagnostic Server API. The other (variable) parameters reappear in the job's parameter profile and the service's DOPs are reused for the job's parameters.

```
<SINGLE-ECU-JOB ID="JOB_Read_Data_By_Address" SEMANTIC="COMMUNICATION">
        <SHORT-NAME>JOB_Read_Data_By_Address</SHORT-NAME>
        <AUDIENCE/>
        <PROG-CODES>
            <PROG-CODE>
                <CODE-FILE>SEJ_ReadDataByAddress</CODE-FILE>
                <SYNTAX>CLASS</SYNTAX>
                <REVISION>2.3.0</REVISION>
            </PROG-CODE>
        </PROG-CODES>
        <INPUT-PARAMS>
            <INPUT-PARAM>
                <SHORT-NAME>Address</SHORT-NAME>
                <DOP-BASE-REF ID-REF="DOP_3ByteIdentical"/>
            </INPUT-PARAM>
            <INPUT-PARAM>
                <SHORT-NAME>Size</SHORT-NAME>
                <DOP-BASE-REF ID-REF="DOP_1ByteIdentical"/>
            </INPUT-PARAM>
        </INPUT-PARAMS>
        <OUTPUT-PARAMS>
            <OUTPUT-PARAM ID="ID_734712">
                <SHORT-NAME>Returned_Value</SHORT-NAME>
                <DOP-BASE-REF ID-REF="DOP_ByteArray"/>
            </OUTPUT-PARAM>
        </OUTPUT-PARAMS>
</SINGLE-ECU-JOB>
```

PROGCODE is extended with an attribute LATEBOUND-CODE-FILE. LATEBOUND-CODE-FILE equals "false" means that the MCD system can load the code file at any time and i.e. keep it in memory until it is needed. If LATEBOUND-CODE-FILE is set to "true" the MCD system loads this job at the latest possible time (which means when the job object is instantiated). In this case, wildcards ('?' or '*') can be used to describe the name of CODE-FILE. If there is exactly one file name matching the wildcard the corresponding job is loaded. If there are multiple files matching the wildcard, the application is asked to provide the file name. If the wildcard does not match any file an error is issued by the MCD system. LATEBOUND-CODE-FILE set to "true" is only allowed for SYNTAX is equal to CLASS or JAR.

7.3.6 DATA PARAMETER

7.3.6.1 OVERVIEW

Data parameters play an important role within diagnostic communication. Diagnostic services send request data to the ECU and get response data back from it. The task of the diagnostic runtime system is not only to perform the communication between tester and ECU but also to transform the data from the physical representation format to the coded byte format that is actually transported via the communication layer and physical layer. The same procedure holds for the reverse way where the coded byte format is converted to the physical representation. The properties of data parameters describe this process of data extraction, interpretation and conversion. The major objects are the DATA-OBJECT-PROPs (or DOPs), which consist of data types for coded and physical format, constraints, computation methods and physical units. The following figure illustrates that:

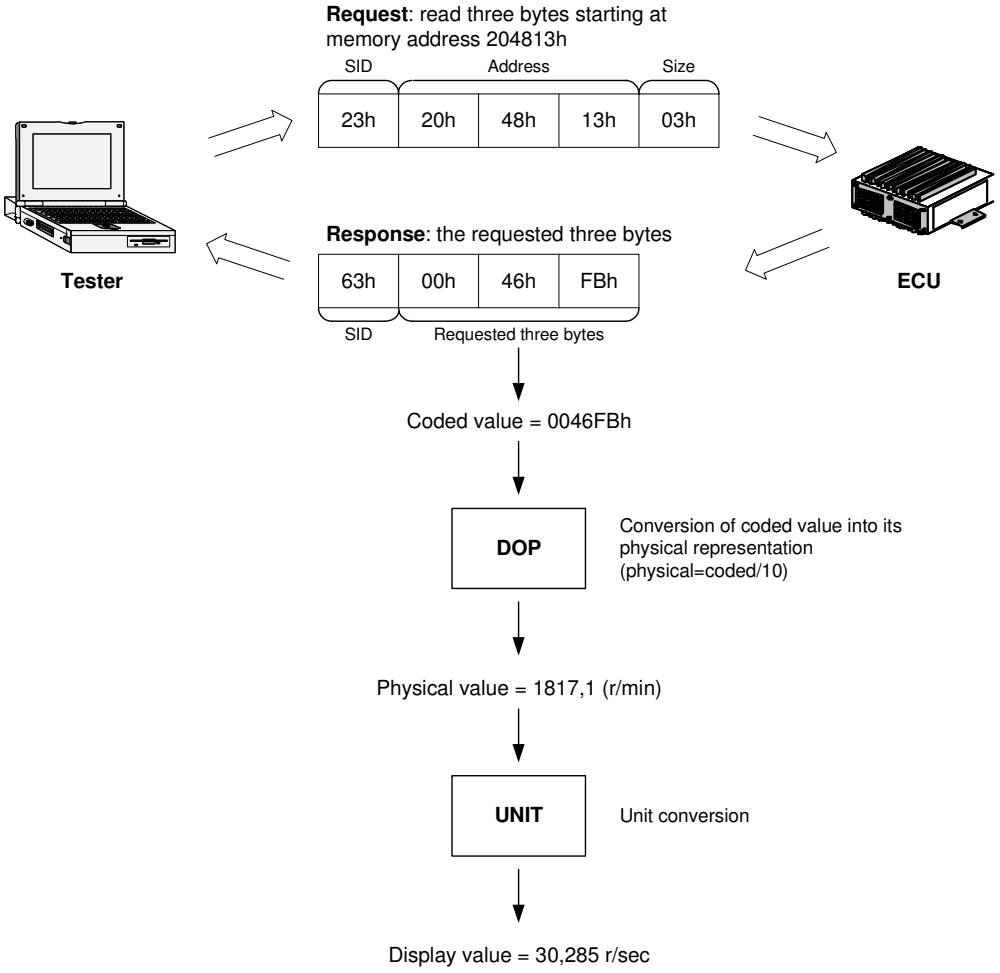


Figure 46 — Data parameters and diagnostic communication (KW2000 example)

Data parameters are a substantial part of the whole ODX data model with a lot of classes and attributes. The following packages keep the model of data parameters:

— **SimpleData** consisting of simple data structures and using the packages

— **Conversion** for the description of transforming data from the physical presentation to the internal format and vice versa: there are 8 different computation method types defined

— **Units** containing the modelling of physical units especially the methods to convert units to each other with consideration of specific unit groups

— **ComplexData** defines composite and diagnostic specific data structures with complex substructures that end up in simple data structures

Drawing: DataParameterPackages
Package: DataParameter

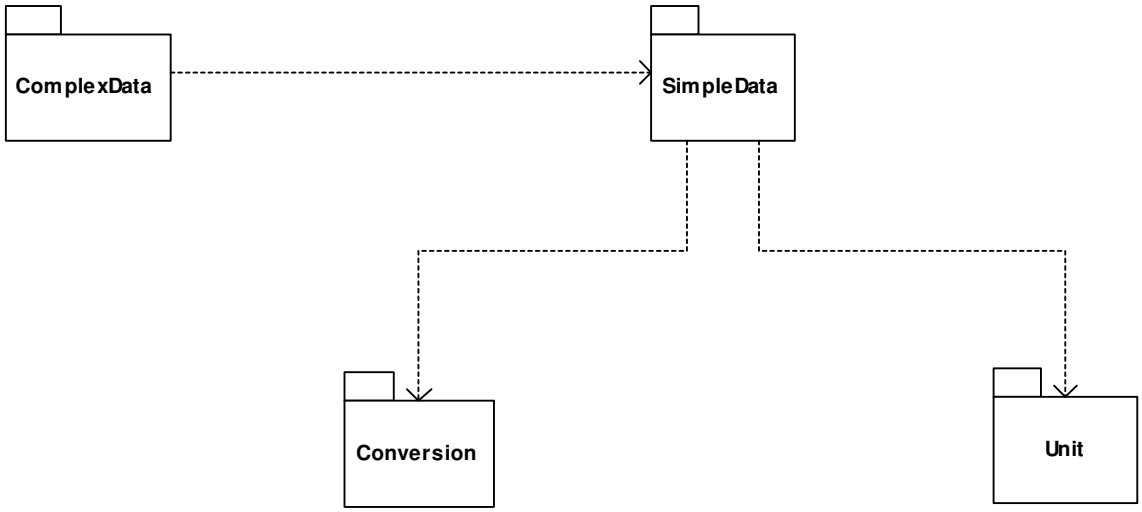


Figure 47 — Data parameter packages

Every DIAG-LAYER can include a data dictionary, which is the location for all diagnostic data parameters. The data dictionary is modelled as class DIAG-DATA-DICTIONARY-SPEC that consists mainly of an aggregation of subclasses defining the different types of data parameters besides administration data and unit definitions.
The following classes of diagnostic data objects are used for request and/or response parameters:

— **DATA-OBJECT-PROP** (SimpleData)**:** consists of data types for coded and physical format, constraints, computation methods and physical units

— **TABLE:** table structure of request and/or response data parameters

— **DTC-DOP:** data parameters for DTCs

— **ENV-DATA-DESC** and **ENV-DATA** (ComplexData): data parameters for environment data belonging to DTC information

— **STRUCTURE** (ComplexData): container for data parameters

— **FIELD (STATIC-FIELD, DYNAMIC-ENDMARKER-FIELD, DYNAMIC-LENGTH-FIELD, END-OF-PDU-FIELD)** (ComplexData): container for data parameters that are repeated within the PDU

— **MUX** (ComplexData): multiplexed data parameters

All used data objects are identified by an ID and have a HANDLE (of type ELEMENT-ID). The general class DOP-BASE is specialized by its subclasses, which are the only ones who can be instantiated (<<impl-child>> inheritance). The most frequently used objects are DATA-OBJECT-PROPs. They are the final classes within an interpretation chain. Exclusively via these objects (and DTC-DOPs) a runtime system is able to deliver values of measured ECU variables or other ECU information to the tester. All other data objects, the COMPLEX-DOPs, describe the packaging rules of special responses from the ECU like repeated structures, multiplex structures or responses with dynamic length. But the description of each COMPLEX-DOP finally ends up with references to DATA-OBJECT-PROPs.

Drawing: DataParameterOverview

Package: DataParameter

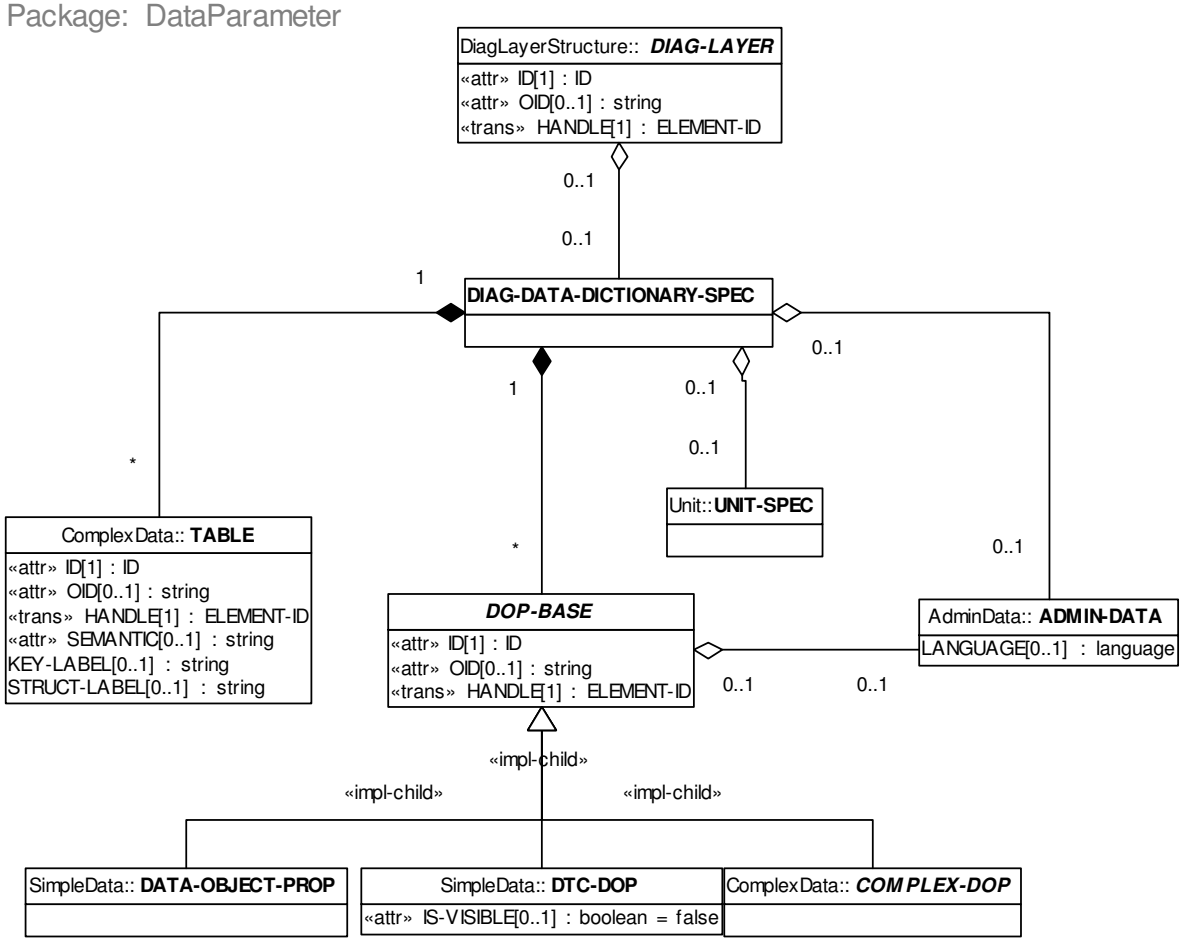


Figure 48 — Data parameter overview

The Conversion package includes all types of computational methods (CompuMethods), which can be used for conversion between physical and coded representation of values. There are eight different types of CompuMethods that are described in the section 7.3.6.6. Where possible, CompuMethods use a generic approach to define the mathematical formula used for conversion. The integration of external computer programs, used for conversion, is also specified in this section.

The package Unit offers the possibility to define units, unit groups and methods for converting units. The class UNIT defines the physical units like Meter, Kilometer, km/h, Seconds and so on with their physical dimensions. UNITs are referenced by DATA-OBJECT-PROPs and indicate to the runtime system that the values from the ECU have to be delivered to the tester inclusive a physical unit (e.g. 100 km/h, 10.3467 s). There is also a concept to define equivalence classes of units used in different countries. For example "mph" is used in the USA where "km/h" is used in Europe. Though "m/s" also represents speed, it is normally used for other values and thus would not belong to this class.
The table below shows the rules for the usage of PARAMs and DOPs. Generally FIELDs must not be contained in FIELDs.

Table 5 — Positioning of PARAM and DOP types

Context	Class	Allowed PARAM types	Allowed DOPs
DIAG-SERVICE	REQUEST	VALUE	DATA-OBJECT-PROP, DTC-DOP, STRUCTURE, END-OF-PDU-FIELD
		CODED-CONST, RESERVED	-
		TABLE-KEY,	TABLE and TABLE-ROW
		TABLE-STRUCT	-
		PHYS-CONST, SYSTEM, LENGTH-KEY	DATA-OBJECT-PROP
DIAG-SERVICE	RESPONSE	VALUE	All DOPs
		CODED-CONST, RESERVED, MATCHING-REQUEST-PARAM, DYNAMIC	-
		PHYS-CONST, SYSTEM, LENGTH-KEY	DATA-OBJECT-PROP
		TABLE-KEY	TABLE
		TABLE-STRUCT	-
REQUEST	STRUCTURE	See "Allowed PARAM types" at REQUEST	See "Allowed DOPs" at REQUEST
RESPONSE	STRUCTURE, ENV-DATA	See "Allowed PARAM types" at RESPONSE	See "Allowed DOPs" at RESPONSE
TABLE-ROW	STRUCTURE	VALUE	all DOPs
		RESERVED	-
		TABLE-KEY	TABLE, TABLE-ROW
		CODED-CONST	-
		PHYS-CONST	DATA-OBJECT-PROP
		SYSTEM	DATA-OBJECT-PROP
		TABLE-ENTRY	TABLE-ROW

7.3.6.2 SIMPLE DATA - DATA OBJECT PROPERTY

A response, which a tester receives from an ECU, can be considered as a byte stream. To interpret this information data object properties (DOP) were introduced. There are two types of DOPs: Simple and complex DOPs. A simple DOP represented by class DATA-OBJECT-PROP describes how to extract a single data item from the byte stream and to transform the extracted (internal) value into its physical representation using a computational method (COMPU-METHOD). After the conversion a UNIT-REF can be used to reference a UNIT in order to provide additional visualisation information for the value. The reverse calculation - i.e. the conversion of the physical value into an internal one and its coding into the byte stream - is also possible with simple DOPs (depending on the used computational method). Complex DOPs are used to structure simple DOPs. They are described in section 7.3.6.10.

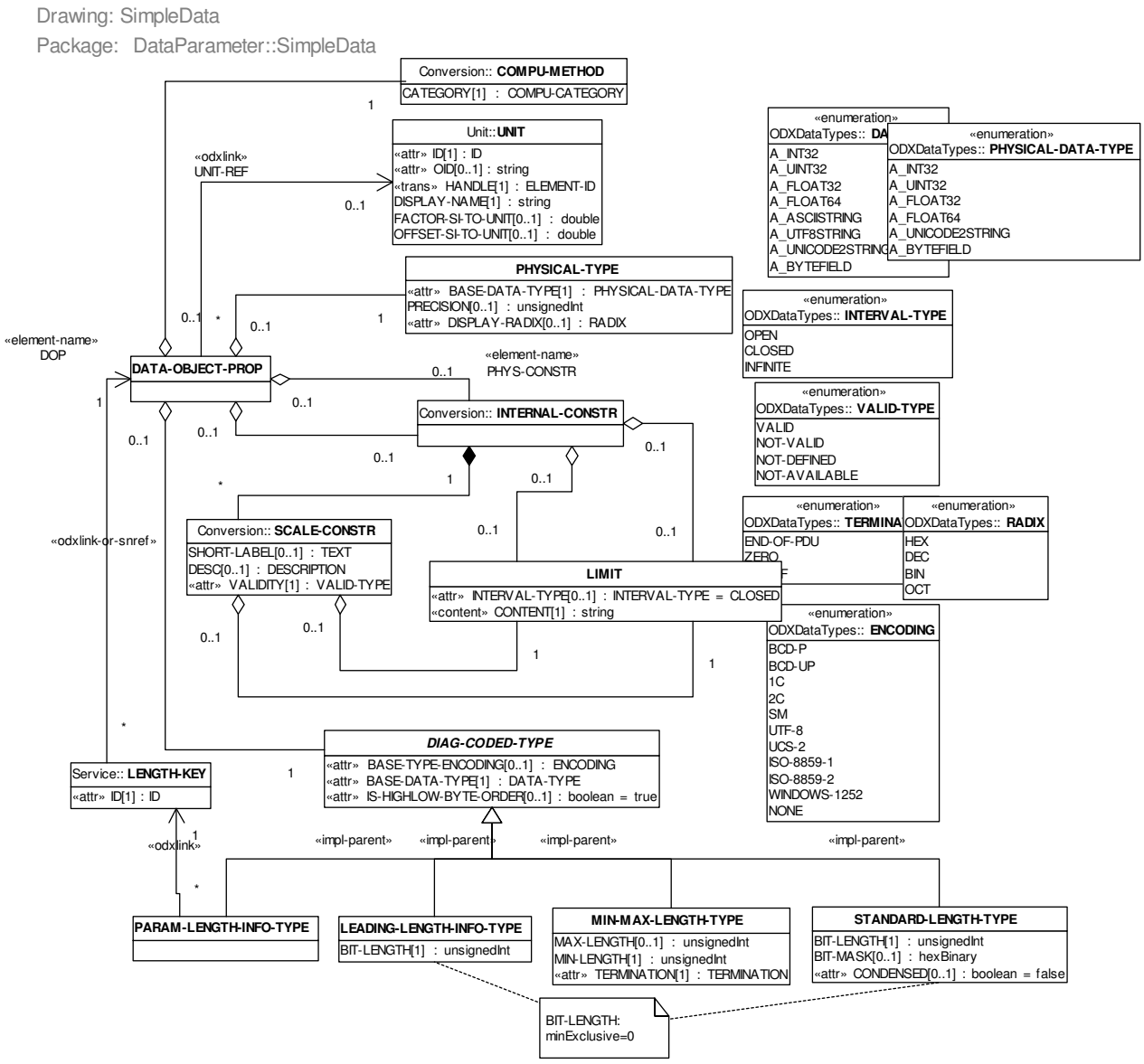


Figure 49 — Simple data

A DOP is identified by its SHORT-NAME or ID. It consists of a COMPU-METHOD, a DIAG-CODED-TYPE and a PHYSICAL-TYPE. Optionally, a UNIT may be referenced and INTERNAL-CONSTR can be given for a DOP.
DIAG-CODED-TYPE is responsible for extraction of the coded value out of the byte stream and for packing of a value into the stream (see sections "Data extraction" and "Data packing" below). The extracted data item is stored within the MCD System as an internal value. This internal value is of a definite type (one of the predefined ODX base data types) that is given by the member BASE-DATA-TYPE of the DIAG-CODED-TYPE.

Table 6 — Predefined ODX base data types

BASE-DATA-TYPE	Description
A_INT32	32 bit signed integer (range: -21474836248 .. 2147483647)
A_UINT32	32 bit unsigned integer (range: 0 .. 4294967295)
A_FLOAT32	32 bit float (IEEE754 single precision)
A_FLOAT64	64 bit float (IEEE754 double precision)
A_ASCIISTRING	ASCII-coded zero-terminated character field (ISO-8859-1) with maximum length: 2^32-1 characters (without delimiter 0x00)
A_UTF8STRING	UTF-8 coded zero-terminated character field with maximum length: 2^32-1 characters (without delimiter 0x00)
A_UNICODE2STRING	UNICODE-2-coded zero-terminated character field (ISO-10646 UCS-2) with maximum length: 2^32-1 characters (without delimiter 0x0000)
A_BYTEFIELD	Byte field with maximum length: 2^32-1

The internal value and its type are used as input for the computational method (COMPU-METHOD) to transform the value into its physical representation (physical value). Consequential the COMPU-METHOD within a DOP must support the internal and physical value type of it. The detailed description of COMPU-METHODs occurs in section 7.3.6.6. There are four types of the class DIAG-CODED-TYPE:

- LEADING-LENGTH-INFO-TYPE: The length of the parameter is to be extracted from the PDU. BIT-LENGTH specifies the bit count at the beginning of this parameter and must be greater than zero. These bits contain the length of the parameter in bytes. The bits of length information are not part of the coded value. That means the data extraction of the coded value starts at the byte edge to this length information. The attribute IS-HIGHLOW-BYTE-ORDER of the DIAG-CODED-TYPE is applied to the extraction of the coded value (exepted A_BYTEFIELD, A_ASCIISTRING, and A_UTF8STRING) and the leading length parameter.

- PARAM-LENGTH-INFO-TYPE: The length of the parameter is defined by another parameter referenced by this kind of DIAG-CODED-TYPE. The physical value of the referenced parameter defines the length of the referencing parameter in bits.

- MIN-MAX-LENGTH-TYPE: The minimum and the maximum length are specified by MIN- and MAX-LENGTH. TERMINATION gives then the termination value, until which the parameter extends. MIN-LENGTH gives the minimal length from which the termination value has to be searched. When the length specified by MAX-LENGTH has been reached (or TERMINATION value is found or the end of the PDU is reached), the searching is stopped and bytes are extracted. The both values (MIN- and MAX-LENGTH) are to be given in bytes. If the actual payload of the parameter is shorter than MAX-LENGTH, the termination value is coded in the PDU. If the actual payload length of the parameter is equal to the MAX-LENGTH, the termination byte is omitted (it is not coded into the request or response message). The termination value in either case is not used for conversion. Only the payload is converted. The length of the data to be extracted is controlled by the value of the member TERMINATION at MIN-MAX-LENGTH-TYPE. The following values for TERMINATION are possible:

  - ZERO: The byte stream to be extracted is terminated by the value 0x00.

  - HEX-FF: The byte stream to be extracted is terminated by the value 0xFF.

  - END-OF-PDU: There is no value that terminates the byte stream. Termination is given by the end of the PDU.

- STANDARD-LENGTH-TYPE: The length of the parameter is specified explicitly by BIT-LENGTH. The value of BIT-LENGTH must be greater than zero. If the BIT-LENGTH bits of a STANDARD-LENGTH-TYPE with BASE-DATA-TYPE of A_ASCIISTRING,

A_UTF8STRING or A_UNICODE2STRING contain a Zero-Byte at the last position (two bytes in case of A_UNICODE2STRING), this is used as the zero-termination of the coded value. If no Zero-Byte is part of the BIT-LENGTH bits, a zero-termination is added to the coded value. If a string value with STANDARD-LENGTH-TYPE is used within a request and the actual data passed by the application as a request parameter yields a shorter length than BIT-LENGTH, the remaining bits have to be filled with 0.

— STANDARD-LENGTH-TYPE is extended with an optional attribute CONDENSED of type Boolean. The default value is “false”. If CONDENSED = true a BIT-MASK is used to special extraction.The semantics of this attribute are as following:

- Attribute CONDENSED = false: There is no impact to the mentioned extraction above.
- Attribute CONDENSED = true. The extraction works like this:

1. BIT-LENGTH parameters are cut out of the PDU at the correct position into a buffer B.
2. The BIT-MASK is applied to these BIT-LENGTH bits in the buffer B (The BIT-MASK needs to contain BIT-LENGTH bits.)
3. All bits of buffer B that are valid through the BIT-MASK (all the ones where the BIT-MASK has a value of 1) are condensed (all other bits removed) and right-aligned.

Example:

With the attribute CONDENSED = true the result of a service execution is

1010**0011** 11000101 00111110 11011011 **0110**1111 .

BYTE-POSITION is 0, BIT-POSITION is 4, BIT-LENGTH is 32, The BIT-MASK is

0000**1111** 00000000 00000000 00000000 **1111**0000

The result of condensing is then **00110110**. Thus the result fits into one byte, instead of five bytes, as with an uncondensed solution.

LEADING-LENGTH-INFO-TYPE and MIN-MAX-LENGTH-TYPE are allowed for the internal data types A_BYTEFIELD, A_ASCIISTRING, A_UNICODE2STRING, and A_UTF8STRING only.
The member ENCODING at DIAG-CODED-TYPE gives the method used for encoding of the value in the PDU. It has the following value domain: BCD-P, BCD-UP, 1C, 2C, SM, UTF-8, UCS-2, ISO-8859-1, ISO-8859-2 and WINDOWS-1252. The following table gives an overview over possible combinations of BASE-DATA-TYPE and BASE-TYPE-ENCODING values.

Table 7 — BASE-DATA-TYPE encodings

BASE-DATA-TYPE	Possible BASE-TYPE-ENCODINGs
A_INT32	SM (sign-magnitude)[2], 1C (one's complement), 2C(two's complement)(=default encoding)[3]
A_UINT32	BCD-P(packed BCD)[4] , BCD-UP[5], NONE (=default encoding)
A_FLOAT32	IEEE754 (=default encoding)
A_FLOAT64	IEEE754 (=default encoding)
A_ASCIISTRING	ISO-8859-1 (=default coding), ISO-8859-2, WINDOWS-1252
A_UTF8STRING	UTF-8 (=default encoding)
A_UNICODE2STRING	UCS-2 (=default encoding)
A_BYTEFIELD	BCD-P(packed BCD), BCD-UP, NONE (=default encoding)

PHYSICAL-TYPE specifies the physical representation of the value. The member BASE-DATA-TYPE gives the type and encoding of the physical value again. Physical values always have the default encoding of their respective type as defined in Table 7. The optional member DISPLAY-RADIX is used for visualisation purposes, i.e. it specifies whether the physical value is to be displayed in a hexadecimal, decimal, binary or octal representation. The corresponding attribute values are HEX, DEC, BIN or OCT. There is also a possibility to specify the number of digits to be displayed after the decimal point by the member PRECISION. It can be given for A_FLOAT32 and A_FLOAT64 data types and is used for the visualisation of the physical value at the tester side. For example, to display the value 1.234 the PRECISION is to be set to 3 while 0.00010 requires a PRECISION of 5. The value coding of numbers in the ODX data is subject to restrictions described by specification "XML Schema Part 2". For example, "100" and "1.0E2" are two different literals which both denote the same float value. A value of type A_BYTEFIELD must be defined hexadecimal in ODX data. Value of type A_UINT32 and A_INT32 must be defined decimal in ODX data.
The calculated physical value can be directly displayed on the tester. Usually units are used to augment this displayed value with additional information like m/s or liter. Also the conversion of values from one unit to another like "km/h" to "m/s" is possible. The declaration of the unit to be used is given via the reference UNIT-REF. See section 7.3.6.7 for more information about units.
The validity of the internal value can be restricted to a given interval via the INTERNAL-CONSTR and its members LOWER-LIMIT and UPPER-LIMIT. Additionally, SCALE-CONSTRS can be used to define valid, non-valid, non-defined or non-available sub-intervals within the interval spanned by INTERNAL-CONSTR. The sub-intervals' limits are defined with LOWER-LIMIT and UPPER-LIMIT in the same way as in INTERNAL-CONSTR. The attribute VALIDITY of each SCALE-CONSTR can take the values VALID, NOT-VALID, NOT-DEFINED, or NOT-AVAILABLE, respectively. Optionally, a SHORT-LABEL can be used to add an identifier to a SCALE-CONSTR.
If no INTERNAL-CONSTR is specified then the ECU handles all possible internal values that are in the range of the internal type.

[2] The encoding sign-magnitude means the first bit represents the sign (e.g. "1" equals "-"). In the following bits the magnitude of the value is coded.

[3] In case the BASE-TYPE-ENCODING is not defined explicitly the "default encoding" has to be applied.

[4] A BCD digit is packed in 4 bits (nibble) (47 = 47h).

[5] A BCD digit is mapped to one byte (17 = 0107h).

Table 8 — Values of VALIDITY

VALIDITY	Description
VALID	Valid area. Current value is within a valid range and can be presented to user.
NOT-VALID	Invalid area. The ECU cannot process the requested data.
NOT-DEFINED	Indicates an area, which is marked in a specification (e.g. as reserved). Shall usually not be set by the ECU but is used by a tester to verify correct ECU behaviour. Also prevents usage of areas during editing process.
NOT-AVAILABLE	Currently invalid area. The value usually is presented by the ECU but can currently not be performed due to e.g. initialisation or temporary problems. This behaviour appears during runtime and cannot be handled while data is edited.

INTERNAL-CONSTR can only be used in conjunction with DOPs, which have a numerical type of the internal value. In case of DOPs used for calculation of physical value from the external one, the internal value is matched against the constraints directly:

1) Match internal value against constraints defined in the INTERNAL-CONSTR of the DOP

2) If match was successful, i.e. the internal value is valid, then calculate the physical value from the internal value with the COMPU-METHOD of the DOP

If a DOP with a defined INTERNAL-CONSTR is used to transform the physical into the internal value, the physical value given by the user is first converted into the internal representation and is then matched against the specified constraints:

3) Calculate the internal value from the physical value with the COMPU-METHOD of the DOP

4) Match calculated internal value against constraints defined in the INTERNAL-CONSTR of the DOP

Perhaps some information like “value is not valid” isn't always sufficient. Therefore the SHORT-LABEL can be used to provide further information like error messages. The attribute INTERVAL-TYPE of LOWER-LIMIT and UPPER-LIMIT defines the type of the interval at the corresponding end. Possible values are OPEN if the edge value is not to be included, CLOSED if it is to be included, or INFINITE if the value defined by LOWER-LIMIT is -∞ or if it is +∞ in the case of UPPER-LIMIT. The value specified by the limit is ignored if INTERVAL-TYPE="INFINITE". In case of no limit given, the interval is not constrained in that direction, i.e. the limit is implicitly set to -∞ if LOWER-LIMIT is missing or to +∞ if no UPPER-LIMIT is given. If a limit with INTERVAL-TYPE="INFINITE" is specified, the value given by this limit is ignored and is set to -∞ or +∞ depending on the interval end.

NOTE The LOWER-LIMIT and UPPER-LIMIT of INTERNAL-CONSTR define the valid interval of values. All values outside of this interval are not valid and thus are not to be processed by the ECU. Therefore, the declaration of an INTERNAL-CONSTR is equivalent to the declaration of a SCALE-CONSTR with VALIDITY="VALID"[6] . If neither LOWER-LIMIT nor UPPER-LIMIT is defined in INTERNAL-CONSTR directly, at least one SCALE-CONSTR with VALIDITY other than "VALID" shall exist.

EXAMPLE Usage of constraints, no restrictions (no INTERNAL-CONSTR), interval (-∞, +∞) is defined as valid

[6] In ODX there is no need for SCALE-CONSTRS with VALIDITY="VALID" because the valid interval is defined by the INTERNAL-CONSTR. If a SCALE-CONSTR with VALIDITY="VALID" is found within the ODX data (e.g. as the result of a data import from another use case) then it should be ignored during the processing of the INTERNAL-CONSTR.

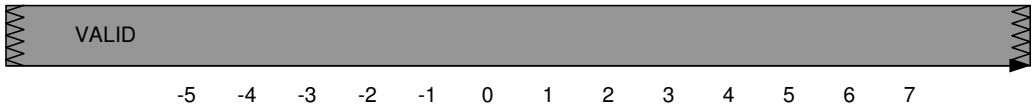


Figure 50 — Usage of constraints: no restrictions

EXAMPLE Usage of constraints. Interval [-3, 5] is defined as valid:

```
<INTERNAL-CONSTR>
       <LOWER-LIMIT INTERVAL-TYPE="CLOSED">-3</LOWER-LIMIT>
       <UPPER-LIMIT INTERVAL-TYPE="CLOSED">5</UPPER-LIMIT>
</INTERNAL-CONSTR>
```

Figure 51 — Usage of constraints: one valid interval

EXAMPLE Usage of constraints, no INTERNAL-CONSTR limits

```
<INTERNAL-CONSTR>
    <SCALE-CONSTRS>
        <SCALE-CONSTR VALIDITY="NOT-DEFINED">
            <LOWER-LIMIT INTERVAL-TYPE="OPEN">4</LOWER-LIMIT>
            <UPPER-LIMIT INTERVAL-TYPE="CLOSED">5</UPPER-LIMIT>
        </SCALE-CONSTR>
        <SCALE-CONSTR VALIDITY="NOT-VALID">
            <LOWER-LIMIT INTERVAL-TYPE="OPEN">5</LOWER-LIMIT>
            <UPPER-LIMIT INTERVAL-TYPE="CLOSED">6</UPPER-LIMIT>
        </SCALE-CONSTR>
    </SCALE-CONSTRS>
</INTERNAL-CONSTR>
```

With no limits for the INTERNAL-CONSTR the validity of the internal value is defined in the interval (-∞, +∞). Within this interval the interval (4, 5] is declared via SCALE-CONSTRS as non-defined and the interval (5, 6] as not valid, which is illustrated in the following figure:

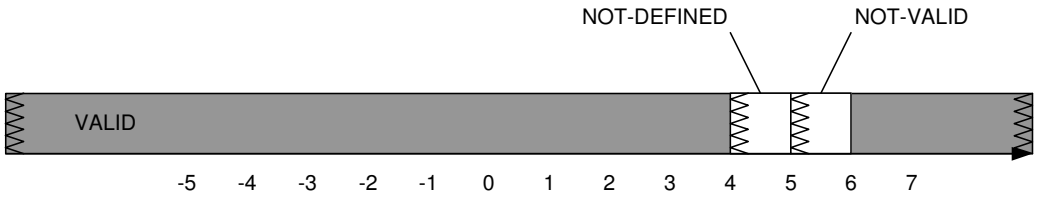


Figure 52 — Usage of constraints: no INTERNAL-CONSTR limits

EXAMPLE Usage of constraints: clipping

```
<INTERNAL-CONSTR>
       <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
       <UPPER-LIMIT INTERVAL-TYPE="INFINITE"/>
       <SCALE-CONSTRS>
              <SCALE-CONSTR VALIDITY="NOT-DEFINED">
                     <LOWER-LIMIT INTERVAL-TYPE="OPEN">4</LOWER-LIMIT>
                     <UPPER-LIMIT INTERVAL-TYPE="CLOSED">5</UPPER-LIMIT>
```

```
            </SCALE-CONSTR>
            <SCALE-CONSTR VALIDITY="NOT-AVAILABLE">
                  <LOWER-LIMIT INTERVAL-TYPE="OPEN">5</LOWER-LIMIT>
                  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">6</UPPER-LIMIT>
            </SCALE-CONSTR>
      </SCALE-CONSTRS>
</INTERNAL-CONSTR>
```

This XML code defines the validity of the internal value in the interval [0, +∞) by defining limits for the INTERNAL-CONSTR. Within this interval the interval (4, 5] is declared via SCALE-CONSTRS as non-defined and the interval (5, 6] as non-available, which is illustrated in the following figure:

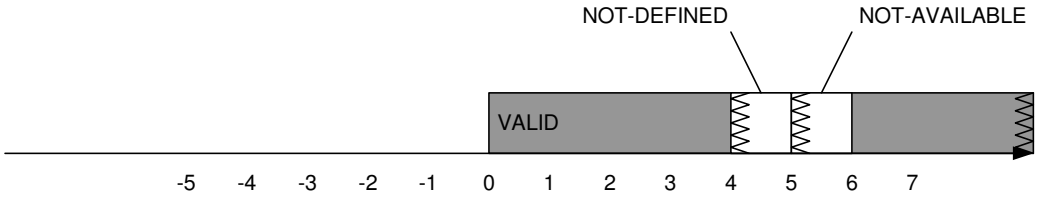


Figure 53 — Usage of constraints: clipping

EXAMPLE Extracting one bit from the PDU and converting it into a string via COMPU-METHOD of type "TEXTTABLE":

```
<DATA-OBJECT-PROP ID="DOP_DTC_WLCS">
      <SHORT-NAME>DOP_DTC_WLCS</SHORT-NAME>
      <LONG-NAME>DTC Warning Lamp Calibration Status</LONG-NAME>
      <COMPU-METHOD>
            <CATEGORY>TEXTTABLE</CATEGORY>
            <COMPU-INTERNAL-TO-PHYS>
                  <COMPU-SCALES>
                        <COMPU-SCALE>
                              <SHORT-LABEL>WLD</SHORT-LABEL>
                              <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-
LIMIT>
                              <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-
LIMIT>
                              <COMPU-CONST>
                                    <VT>warning lamp disabled</VT>
                              </COMPU-CONST>
                        </COMPU-SCALE>
                        <COMPU-SCALE>
                              <SHORT-LABEL>WLE</SHORT-LABEL>
                              <LOWER-LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-
LIMIT>
                              <UPPER-LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-
LIMIT>
                              <COMPU-CONST>
                                    <VT>warning lamp enabled</VT>
                              </COMPU-CONST>
                        </COMPU-SCALE>
                  </COMPU-SCALES>
            </COMPU-INTERNAL-TO-PHYS>
      </COMPU-METHOD>
      <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32"
IS-HIGHLOW-BYTE-ORDER="true">
            <BIT-LENGTH>16</BIT-LENGTH>
      </DIAG-CODED-TYPE>
      <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
      <INTERNAL-CONSTR>
            <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
            <UPPER-LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
```

```
      </INTERNAL-CONSTR>
</DATA-OBJECT-PROP>
```

EXAMPLE Extracting two bytes from the PDU and calculating the vehicle velocity in km/h:

```
<DATA-OBJECT-PROP ID="DOP_VehicleVelocity">
      <SHORT-NAME>DOP_VehicleVelocity</SHORT-NAME>
      <COMPU-METHOD>
            <CATEGORY>IDENTICAL</CATEGORY>
      </COMPU-METHOD>
      <DIAG-CODED-TYPE BASE-DATA-TYPE="A_INT32" xsi:type="STANDARD-LENGTH-TYPE">
            <BIT-LENGTH>16</BIT-LENGTH>
      </DIAG-CODED-TYPE>
      <PHYSICAL-TYPE BASE-DATA-TYPE="A_INT32"/>
      <UNIT-REF ID-REF="Unit_Kmh"/>
</DATA-OBJECT-PROP>
```

where Unit_Kmh and the appropriate physical dimension are defined as follows:

```
<UNIT ID="Unit_Kmh">
      <SHORT-NAME>Unit_Kmh</SHORT-NAME>
      <DISPLAY-NAME>km/h</DISPLAY-NAME>
      <FACTOR-SI-TO-UNIT>3.6</FACTOR-SI-TO-UNIT>
      <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-UNIT>
      <PHYSICAL-DIMENSION-REF ID-REF="PD_Velocity"/>
</UNIT>

<PHYSICAL-DIMENSION ID="PD_Velocity">
      <SHORT-NAME>PD_Velocity</SHORT-NAME>
      <LENGTH-EXP>1</LENGTH-EXP>
      <TIME-EXP>-1</TIME-EXP>
</PHYSICAL-DIMENSION>
```

7.3.6.3 DATA EXTRACTION

For the data extraction process of PARAM from the PDU to the physical value the following data objects are used:

- BYTE-POSITION and BIT-POSITION at PARAM which references this DATA-OBJECT-PROP,
- DIAG-CODED-TYPE,
- PHYSICAL-TYPE,
- COMPU-METHOD,
- IS-HIGHLOW-BYTE-ORDER,
- BIT-MASK,
- ENCODING.

The data extraction process comprises the following steps:

p) Extraction of ByteCount bytes, starting at PARAM /BYTE-POSITION, where ByteCount is computed as follows:

$$ByteCount = \left\lceil \frac{BitPosition + BitLength}{8} \right\rceil$$, where $\lceil \ \rceil$ indicates the ceiling function.

q) Normalisation of data: If IS-HIGHLOW-BYTE-ORDER="false" (Intel format), the extracted bytes are to be reversed, so that the byte order corresponds to the Motorola format (HIGHLOW). This step is only applied if coded type represents a numeric value.

r) Extraction of BIT-LENGTH bits from the normalized byte field starting at PARAM /BIT-POSITION. The valid value range of BIT-POSITION is 0 to 7. Bit counting starts with the least significant (right-most) bit of the least significant byte where BIT-POSITION is equal 0. If no BIT-POSITION is specified, the value of 0 is taken for it. BIT-LENGTH in the case of a MIN-MAX-LENGTH-TYPE is the netto bitlength neglecting a possible termination byte.

s) Appliance of the BIT-MASK as AND-mask without any implicit shift operation.

t) Assignment of coded value using of ENCODING in order to interpret the extracted bit field with the length BIT-LENGTH.

u) Computation of the physical value using COMPU-METHOD.

These steps have to be performed in the reverse direction in the case of a request.
The goal of the first 5 steps is to internally assign the received data to a variable of coded type. This variable is used in the last step to calculate the physical value using the specified COMPU-METHOD. The result is a variable of type PhysicalType holding the physical value corresponding to the received coded value. This physical value is finally displayed to the user (potentially, after converting to desired units).
As an example, the following figure illustrates the extraction process of a 4-bit value from the PDU:

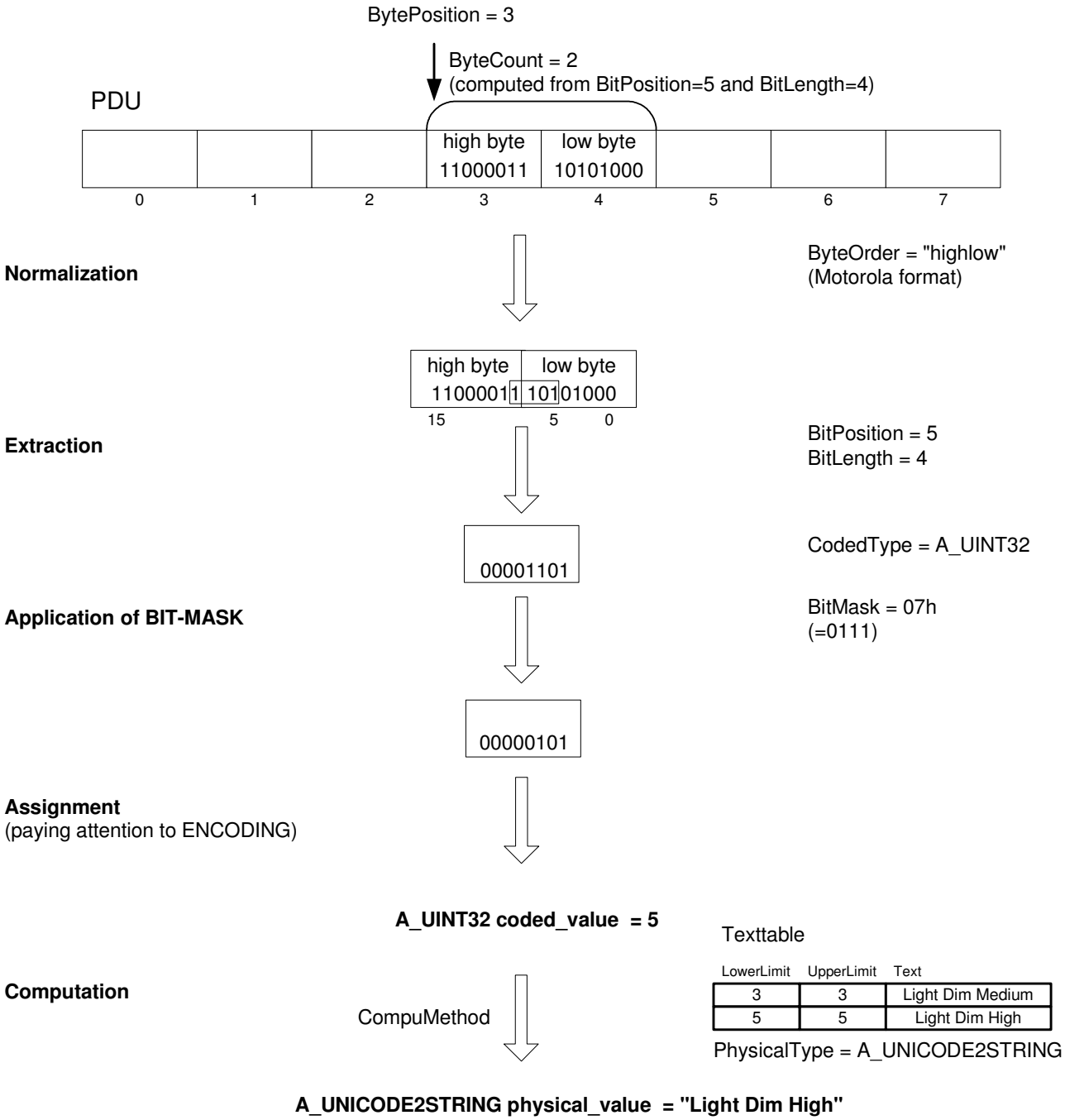


Figure 54 — Example: Extraction of a 4-bit value

At first, the value ByteCount is computed:

$$ByteCount = \left\lceil \frac{BitPosition + BitLength}{8} \right\rceil = \left\lceil \frac{5+4}{8} \right\rceil = 2$$

A two-byte sub-frame is then extracted from the original data frame consisting of the bytes 3 and 4 (BytePosition = 3, the ByteCount = 2). Since ByteOrder="highlow" the normalisation step is not necessary. In the next step, the relevant bits are extracted from the two-byte frame using BitPosition and BitLength. Now the BitMask is applied and the value is assigned to a variable of the specified CodedType. At last, the CompuMethod (in

this case a texttable) is applied to the coded value and the result is assigned to a new string-valued variable.

7.3.6.4 DATA PACKING

The process of data packing, i.e. the conversion of the physical value into its internal representation and putting it into the PDU, takes the following steps:

v) Value preparation. The physical value is prepared in order to be put into the PDU. The following steps are necessary:

1) Computation using COMPU-METHOD: the physical value is converted into the internal representation. The bytes of the target value buffer are set to zero by default.

2) Creation of bit-stream using ENCODING

3) Determine the effective BitLength for the coded data types A_BYTEFIELD, A_ASCIISTRING, A_UTF8STRING, and A_UNICODE2STRING including termination character if necessary.

4) Application of BIT-MASK if this is specified: a bit-wise AND-operation is applied to the bit-stream and the BIT-MASK. The BIT-MASK here is used to manipulate the value that comes out of the computing method.

5) Truncation the coded value according to BitLength: all bit positions in the bit-stream above BitLength will be cleared.

w) Preparation of a PDU cutout:

1) Extraction of ByteCount bytes, starting at PARAM /BYTE-POSITION, where ByteCount is computed as follows:

$$ByteCount = \left\lceil \frac{BitPosition + BitLength}{8} \right\rceil$$

If the bytes do not exist at the time, the parameter is to be processed, they are created and initialized with 0. The result of this step is a PDU cutout with the length of ByteCount bytes.

2) Normalisation: if IS-HIGHLOW-BYTE-ORDER="false" (Intel format), the extracted bytes are to be reversed, so that the byte order corresponds to the Motorola format (HIGHLOW). This step is only applied if coded type represents a numeric value. In the case of UNICODE string, each pair of bytes, representing a single character, is swapped separately.

3) Application of BIT-MASK: the bit in the bit-stream is set to 0, if the corresponding bit in the BIT-MASK is set to 1 (i. e. clear all bits, which have the value of 1 in the BIT-MASK). The bit 0 of the BIT-MASK is applied to the bit x in the PDU cutout, where x=BitPosition; bit 1 is applied to the bit x+1 and so on. The default BIT-MASK if not specified is always a sequence of bit "1" according to the length BitLength. In other words: If BIT-MASK is not defined the bit-stream has to be copied into the PDU limited by BitLength.

NOTE BIT-MASK is used to protect bits in the PDU from manipulation, e.g. because they are already set by other PARAMs.

x) Merging: the BitLength bits of the PDU cutout and the bit-stream are merged by applying a logical OR-operation. The bit 0 of the bit-stream is applied to the bit x in the PDU cutout, where x=BitPosition; bit 1 is applied to the bit x+1 and so on.

y) Packing into the PDU:

1) Normalisation: if IS-HIGHLOW-BYTE-ORDER="false" (Intel format), the extracted bytes are to be reversed again to cancel the first normalisation out.

2) Put back the PDU cutout into the PDU.

The following figure exemplifies the step 1):

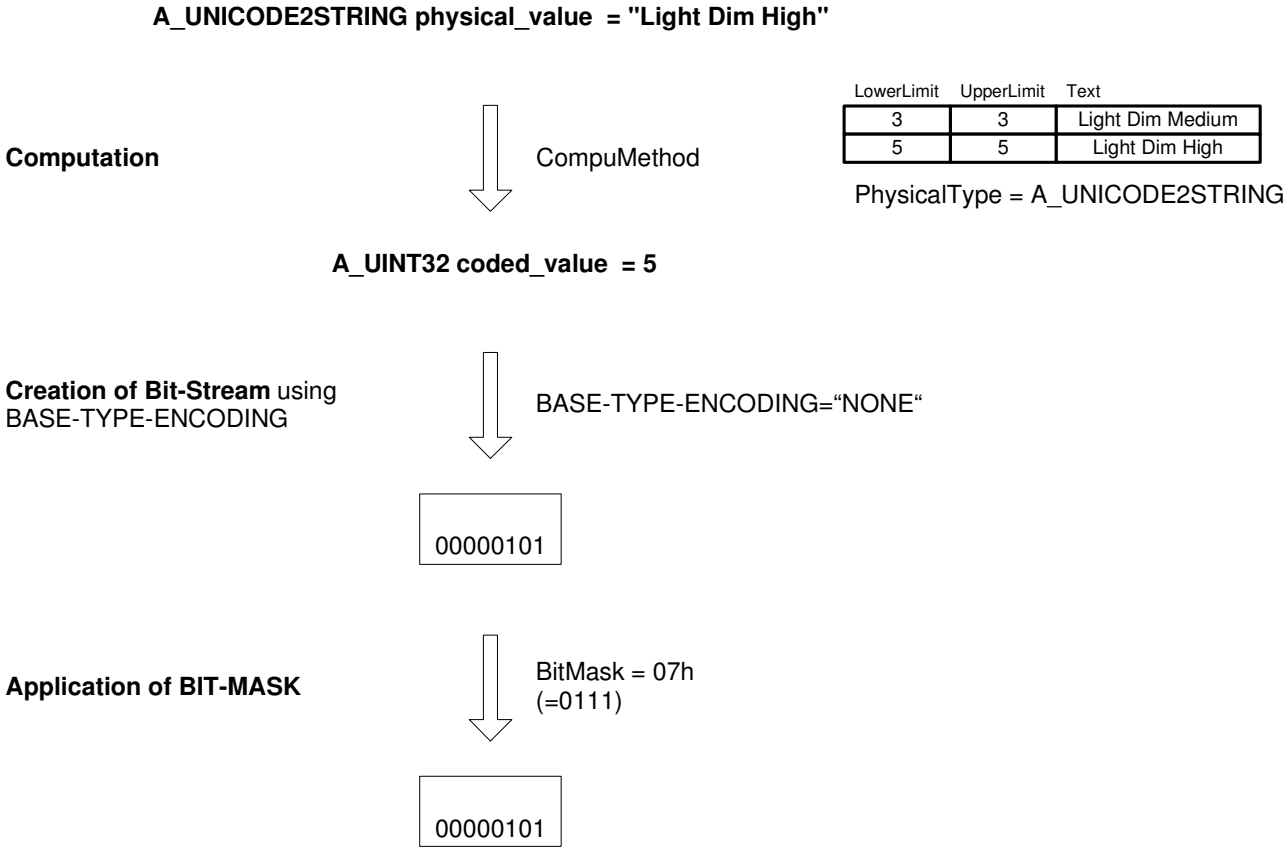


Figure 55 — Example: preparation of physical value for data packing

The steps 2), 3) and 4) can be visualized as follows:

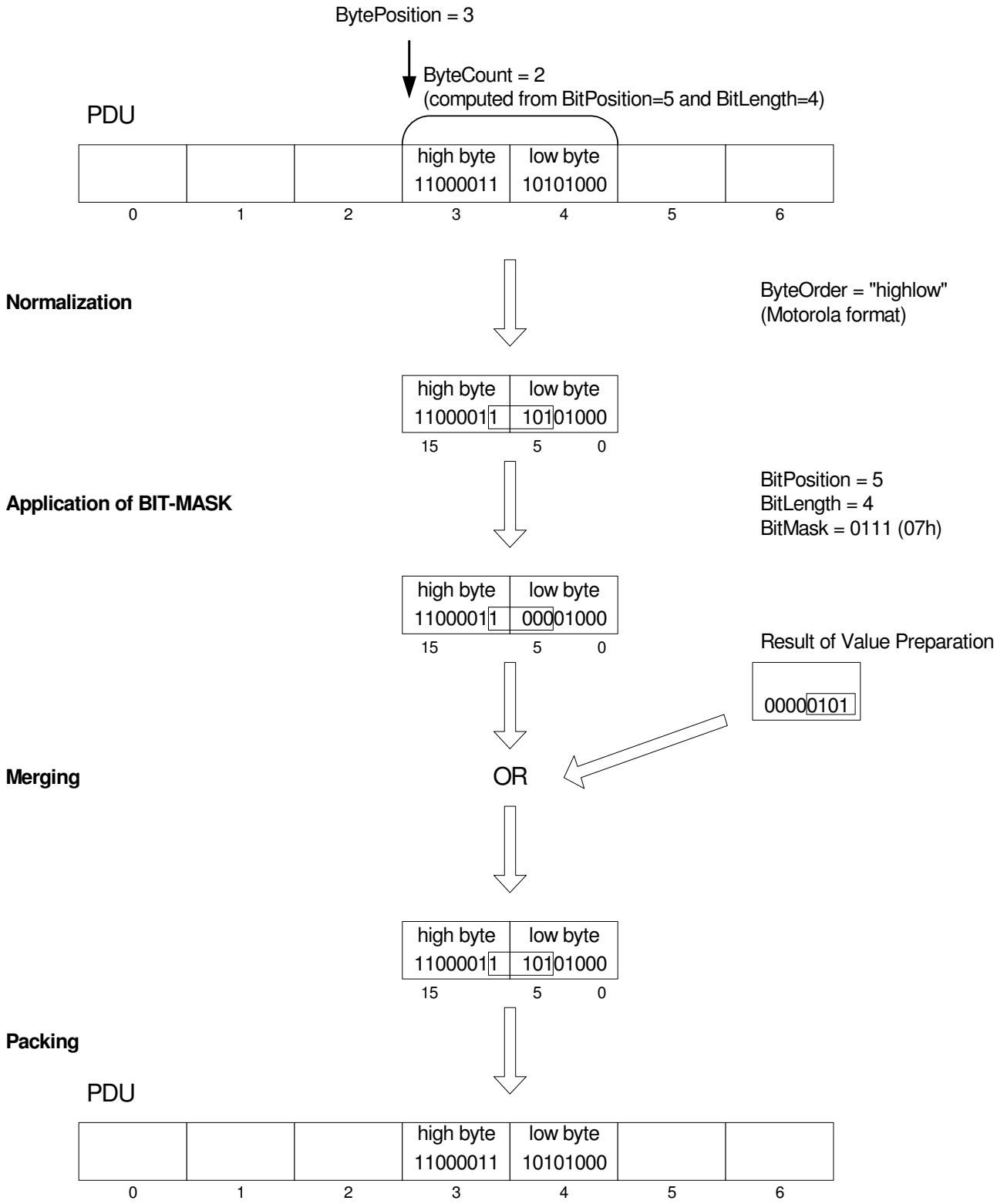


Figure 56 — Example: data packing

The following items should be pointed out:

z) If only whole bytes are affected, e.g. for coded types A_BYTEFIELD, A_ASCIISTRING, A_UTF8STRING, and A_UNICODE2STRING without BIT-MASK and BIT-POSITION, the algorithm described above is equivalent to a simple coding and copying of the value into the

PDU starting at the BYTE-POSITION. In the case of A_UNICODE2STRING, the flag IS-HIGHLOW-BYTE-ORDER="false" has to be considered for each pair of bytes representing a single UNICODE character.

aa) The optional BIT-MASK specifies the bits inside the PDU which are changed by the value. The steps 1 iv), 1 v), 2 iii) and 3) are equivalent to the rule, that only those bits of the bit-stream are moved (copied) into the PDU, those corresponding bit in the bitmask are set to 1 and all other bits in the PDU remains unchanged.

bb) The data extraction is the logical inversion of the data packing. Some steps are not required for data extraction because the PDU cutoff will not put back into the PDU or the applying of the BIT-MASK in steps 1 iv) and 2 iii) are combined in only one step.

cc) The optional leading length parameter has to be put separately from the other parameter like an own request parameter.

7.3.6.5 VALUE COMPARISON

Definition of LOWER- and UPPER-LIMITs in COMPU-SCALEs, INTERNAL-CONSTRs and SCALE-CONSTRs requires the definition of comparision operators "<" (less than), "==" (equals) and ">" (greater than) for each data type.
In case of A_UINT32, A_INT32, A_FLOAT32, A_FLOAT64 the values are compared numerically.
In case of A_ASCIISTRING, A_UNICODE2STRING and A_UTF8STRING only the comparison operator "equals" is defined. (LOWER- and UPPER-LIMIT must be equal or UPPER-LIMIT must not be defined meening that it is equal to LOWER-LIMIT.
In case of A_BYTEFIELD the comparison operators are defined as follows. The values are filled up with 00-Bytes until they are of same length. Right-most byte is the least significant byte. After that the values are interpreted as numerical values (large unsigned integers) and can be compared like other numerical types. For example, the following (in-)equations are valid for byte fields:
FF01 > 00AE02; 00FF01 = FF01; 00AE02 < FF01; FF01 = 00FF01

7.3.6.6 COMPUTATIONAL METHODS

7.3.6.6.1 General

Computational methods (COMPU-METHOD) are needed to calculate between the internal type and the physical type of the diagnostic values. An internal diagnostic value is for example the result of the extraction of a response parameter from the response message received from the ECU. To this internal diagnostic value a computational method is applied to get the physical value for representation. On the other hand a physical value given by the user is to be converted into internal value in order to be sent to an ECU. This is also done by means of the CompuMethods. The extraction of data items from a response and the coding of data items into a request are made via DIAG-CODED-TYPE that is described in section 7.3.6.2.

Drawing: Conversion
Package: DataParameter::Conversion

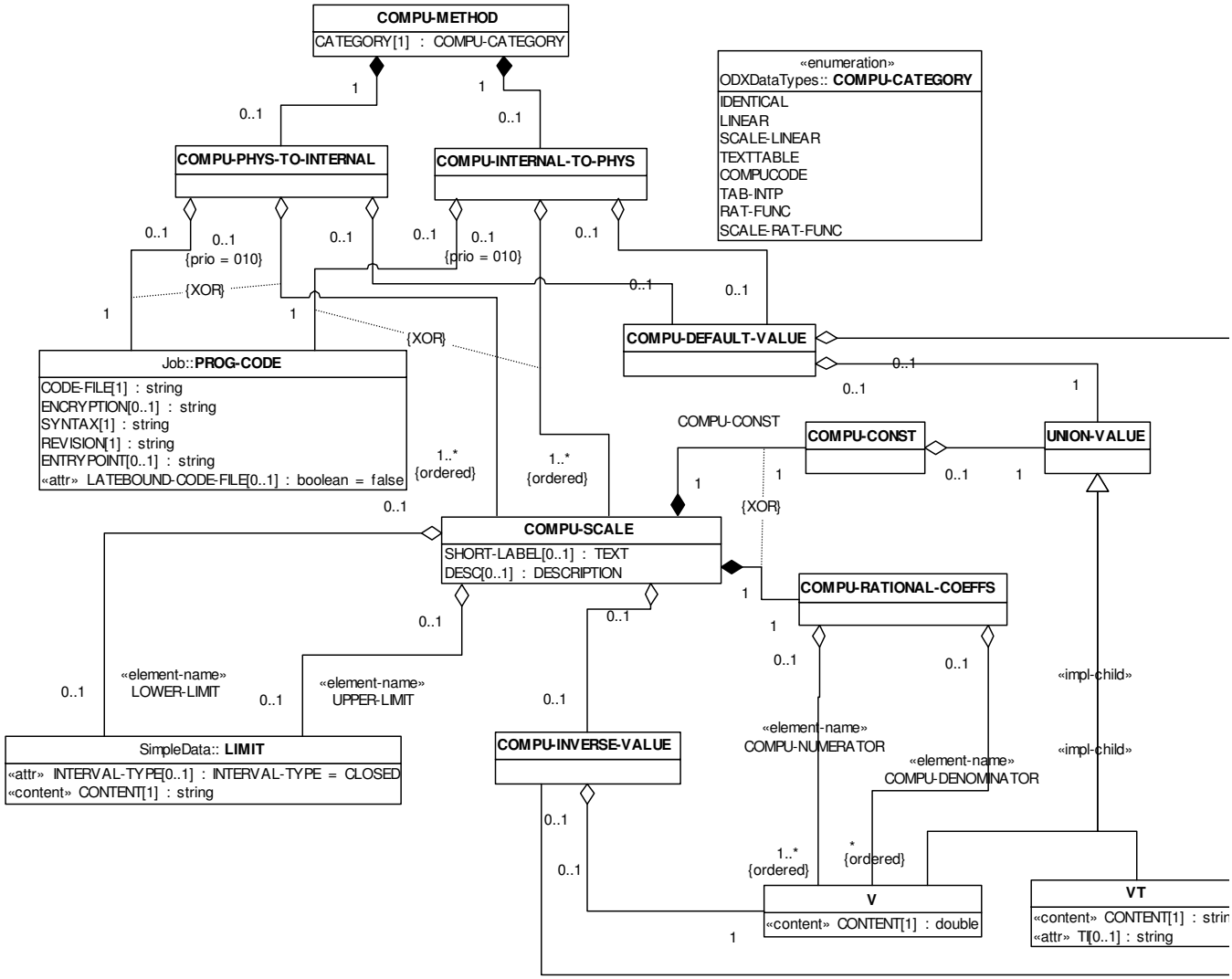


Figure 57 — Conversion

A COMPU-METHOD can either specify a COMPU-INTERNAL-TO-PHYS or a COMPU-PHYS-TO-INTERNAL element, which specifies the transformation of a coded value read out of the PDU into a physical value of a type compliant to the DOP-BASE this COMPU-METHOD belongs to or a physical value into a coded value, respectively. For invertible COMPU-METHODS (e.g. a LINEAR function), the inversion has to be performed automatically by e.g. a MCD 3D / MVCI Diagnostic Server system. In case the COMPU-INTERNAL-TO-PHYS calculation is not invertible the COMPU-INVERSE-VALUE can be used or the corresponding COMPU-PHYS-TO-INTERNAL element can be used to define a deterministic inverse calculation. The same holds true, if only a COMPU-PHYS-TO-INTERNAL has been defined at this is not invertible. A COMPU-INTERNAL-TO-PHYS can be used to solve this situation. In case either of the two elements is of CATEGORY COMPU-CODE, it is never invertible and a separate element for the inverse direction has to be specified. Be aware that the author is responsible for the consistency between the

COMPU-INTERNAL-TO-PHYS and the COMPU-PHYS-TO-INTERNAL calculation. There is no possibility for an ODX tool to check that the one is really a valid inversion of the other one.
For an example on the usage of COMPU-INVERSE-VALUE please refer to section 6.3.6.3.6, where a TEXTTABLE is specified. Since multiple internal values are mapped onto the same string as physical value, there is no deterministic inversion. However, for every COMPU-SCALE element a COMPU-INVERSE-VALUE is specified. For example, the values 5..9 map onto the string "ON". Since it is non-deterministic to which value "ON" should be mapped when converting from physical to internal value, COMPU-INVERSE-VALUE defines this deterministically to be the value 5.
In case where both directions (COMPU-PHYS-TO-INTERNAL and COMPU-INTERNAL-TO-PHYS) are existing in one COMPU-METHOD both need to be used in case this COMPU-METHOD appears in a REQUEST and RESPONSE. If only one direction is used, inversion takes place in the respective other direction, if the corresponding conversion is invertible.
There are eight different types of CompuMethods: IDENTICAL, LINEAR, SCALE-LINEAR, TEXTTABLE, COMPUCODE, TAB-INTP, RAT-FUNC and SCALE-RAT-FUNC. The mandatory member CATEGORY defines the type of the CompuMethod.
A generic approach is used for the definition of functions. The following mathematical function forms the basis of all functions apart from IDENTICAL, TEXTTABLE and COMPUCODE:

$$f(x) = \frac{V_{N0} + V_{N1} * x + V_{N2} * x^2 + \ldots + V_{Nn} * x^n}{V_{D0} + V_{D1} * x + V_{D2} * x^2 + \ldots + V_{Dm} * x^m}$$

Numerator
Denominator

For each interval a numerator (COMPU-NUMERATOR) and a denominator (COMPU-DENOMINATOR) are defined. Each of them defines a sequence of V-values representing a polynomial. The first V is the coefficient for x^0, the second V is the coefficient for x^1 and so on.
The domain of TEXTTABLE, SCALE-LINEAR and SCALE-RAT-FUNC functions (and only of these functions) can be divided into several intervals. The domain of other functions (all functions apart from TEXTTABLE, SCALE-LINEAR and SCALE-RAT-FUNC) must not be divided into intervals. Just a single COMPU-SCALE is to be defined within COMPU-SCALES in this case. Identical functions do not allow the definition of scales at all.
A scale for a function is defined via COMPU-SCALE. When applying intervals for scales, defined by UPPER-LIMIT and LOWER-LIMIT, the user has to guarantee that no intervals will overlap. Therefore the attribute INTERVAL-TYPE and its values "OPEN", "CLOSED" or "INFINITE" can be used. The value "OPEN" does not include the boundary, whereas the value "CLOSED" does. "INFINITE" denotes an open-end interval. LOWER-LIMIT must be defined in any case of a COMPU-SCALE definition. If no UPPER-LIMIT is defined the COMPU-SCALE will be applied only for the value defined in LOWER-LIMIT. In this case INTERVAL-TYPE at LOWER-LIMIT must be defined as CLOSED.
If the internal value does not match any defined interval (wherever they can be defined) the runtime system has to indicate a runtime error.
For all CompuMethods except of IDENTICAL, TEXTTABLE and COMPUCODE the calculation type (float or int) is deduced from the type of the internal and physical values. It has a decisive influence on the precision of the calculation. The type of the calculation is determined according to the following rules:

dd) If one of the types (physical or internal) is FLOAT (A_FLOAT32 or A_FLOAT64), the calculation is of type FLOAT (64 bit).

ee) If both types are UINT (A_UINT32), the calculation is of type UINT.

ff) In any other case, the calculation is of type INT.

The type of the coefficients in NUMERATOR is equal to the type of the calculation. The type of coefficients in DENOMINATOR is UINT. The DENOMINATOR must be non-zero. In case of TAB-INTP the type of LOWER-LIMIT is equal to the internal type, and the type of COMPU-CONST is equal to the physical type. If an overflow or division by zero occurs during the calculation, the runtime system has to report an error message.
In the following all of the eight CompuMethod types are described in detail.

7.3.6.6.2 Identical

A CompuMethod of type IDENTICAL (CATEGORY="IDENTICAL") just passes on the input value so that the internal value and the physical one are identical.

Mathematical function:

$$f(x) = x, x \in R$$

Certain data types:
Internal: all types
Physical: all types
(physical type = internal type)

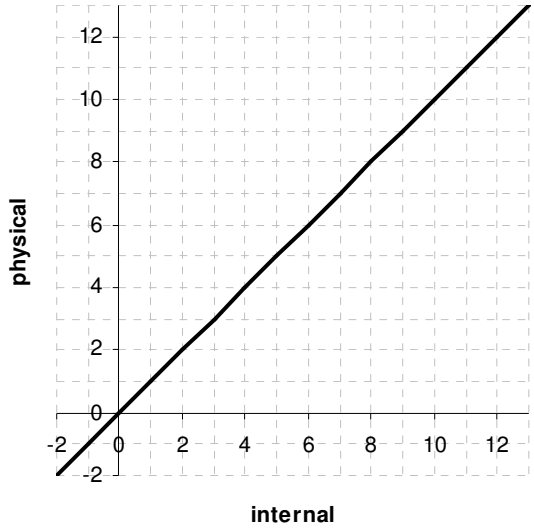


Figure 58 — Identical

For CompuMethods of this type the data objects COMPU-INTERNAL-TO-PHYS and COMPU-PHYS-TO-INTERNAL are not allowed. This is the simplest type of a CompuMethod and can be instantiated as follows:
EXAMPLE

```
<COMPU-METHOD>
      <CATEGORY>IDENTICAL</CATEGORY>
</COMPU-METHOD>
```

The value domain of an identical function includes all values of all types (including ascii strings). This does not become apparent from the drawing above, but it is possible. The only limitation, which must be observed, is that the physical and the internal types must be identical excepting A_ASCIISTRING and A_UTF8STRING at BASE-TYPE-ENCODING at DIAG-CODED-TYPE. In this case the conversion to physical BASE-DATA-TYPE A_UNICODE2STRING must be done during computational process.

7.3.6.6.3 Linear

CompuMethods of type LINEAR multiply the internal value with a factor and add on an offset. Exactly one COMPU-SCALE has to be defined. It contains COMPU-RATIONAL-COEFFS within which COMPU-NUMERATOR and COMPU-DENOMINATOR are declared. The numerator should contain two values. The first one is the offset (V_{N0}), the

second one is the factor (V_{N1}). The denominator must have an integer value.
The LIMITs at COMPU-SCALE can be used to restrict the value domain.
NOTE If physical data type is A_FLOATxx and internal data type is A_INT32 or A_UINT32 the following rounding is applied: 0.5 -> 1, 1.3 -> 1, 1.5 ->2.

Via this CompuMethod it is possible to create a function returning a constant value by setting $V_{N1} = 0$. In this case COMPU-INVERSE-VALUE has to be defined if the reverse calculation is required.

Mathematical function:

$$f(x) = \frac{V_{N0} + V_{N1} * x}{V_{D0}}$$

Possible data types:
Internal: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64
Physical: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64

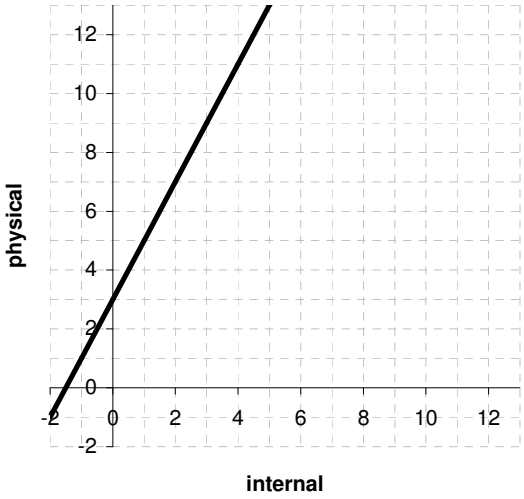


Figure 59 — Linear

EXAMPLE

$$f(x) = 3 + 2x$$

```
<COMPU-METHOD>
       <CATEGORY>LINEAR</CATEGORY>
       <COMPU-INTERNAL-TO-PHYS>
              <COMPU-SCALES>
                     <COMPU-SCALE>
                            <COMPU-RATIONAL-COEFFS>
                                   <COMPU-NUMERATOR><V>3</V><V>2</V></COMPU-
NUMERATOR>
                                   <COMPU-DENOMINATOR><V>1</V></COMPU-DENOMINATOR>
                            </COMPU-RATIONAL-COEFFS>
                     </COMPU-SCALE>
              </COMPU-SCALES>
       </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

7.3.6.6.4 Scale linear

Scale linear functions (CATEGORY="SCALE-LINEAR") are similar to the linear ones with the difference that more than one interval can be defined as the domain of the function and different calculations can be applied to the intervals. The boundaries of the intervals must not overlap. The calculation in the reverse direction (physical to internal) is allowed for this type. If VN1=0 COMPU-INVERSE-VALUE may be used to specify a unique reverse calculation. If COMPU-INVERSE-VALUE is not specified the first coded value of the scale is used for the reverse calculation in this case.
Condition for invertibility: Adjacent COMPU-SCALES must have the same values on their common boundaries AND the VN1 of all intervals must have the same sign or must be 0. If VN1 = 0 then COMPU-INVERSE-VALUE must be specified.

Mathematical function:

$$f(x) = \begin{cases} \dfrac{V_{N0}^1 + V_{N1}^1 * x}{V_{D0}^1}, x \in I_1 \\ \cdots \\ \dfrac{V_{N0}^n + V_{N1}^n * x}{V_{D0}^n}, x \in I_n \end{cases}$$

where I_1, … , I_n are arbitrary disjunctive intervals.

Possible data types:
Internal: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64
Physical: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64

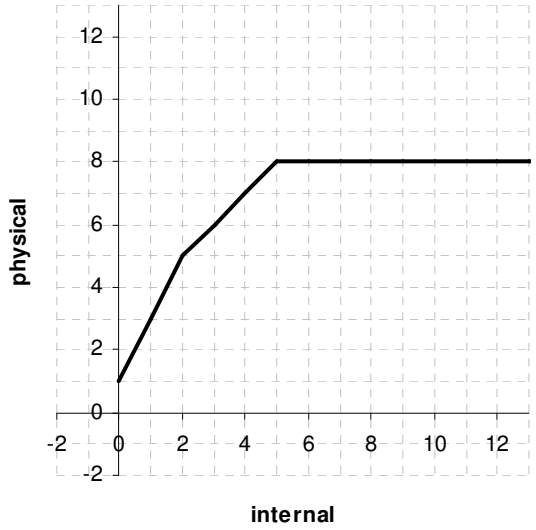


Figure 60 — Scale linear

EXAMPLE

$$f(x) = \begin{cases} 1 + 2x, x \in [0,2) \\ 3 + x, x \in [2,5) \\ 8, x \in [5,+\infty) \end{cases}$$

```
<COMPU-METHOD>
       <CATEGORY>SCALE-LINEAR</CATEGORY>
       <COMPU-INTERNAL-TO-PHYS>
              <COMPU-SCALES>
                     <COMPU-SCALE>
                            <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
                            <UPPER-LIMIT INTERVAL-TYPE="OPEN">2</UPPER-LIMIT>
                            <COMPU-RATIONAL-COEFFS>
                                   <COMPU-NUMERATOR><V>1</V><V>2</V></COMPU-
NUMERATOR>
                                   <COMPU-DENOMINATOR><V>1</V></COMPU-DENOMINATOR>
                            </COMPU-RATIONAL-COEFFS>
                     </COMPU-SCALE>
                     <COMPU-SCALE>
                            <LOWER-LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
                            <UPPER-LIMIT INTERVAL-TYPE="OPEN">5</UPPER-LIMIT>
```

```
                        <COMPU-RATIONAL-COEFFS>
                              <COMPU-NUMERATOR><V>3</V><V>1</V></COMPU-
NUMERATOR>
                              <COMPU-DENOMINATOR><V>1</V></COMPU-DENOMINATOR>
                        </COMPU-RATIONAL-COEFFS>
                  </COMPU-SCALE>
                  <COMPU-SCALE>
                        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">5</LOWER-LIMIT>
                        <UPPER-LIMIT INTERVAL-TYPE="INFINITE"/>
                        <COMPU-RATIONAL-COEFFS>
                              <COMPU-NUMERATOR><V>8</V><V>0</V></COMPU-
NUMERATOR>
                              <COMPU-DENOMINATOR><V>1</V></COMPU-DENOMINATOR>
                        </COMPU-RATIONAL-COEFFS>
                  </COMPU-SCALE>
            </COMPU-SCALES>
      </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

7.3.6.6.5 Rational function

A rational function (CATEGORY="RAT-FUNC") differs from the linear function in the omission of the restrictions for COMPU-NUMERATORs and COMPU-DENOMINATORs. They can have as many V-values as needed. The calculation in the reverse direction (physical to internal) is not allowed for this type.

Mathematical function:

$$f(x) = \frac{V_{N0} + V_{N1} * x + V_{N2} * x^2 + ... + V_{Nn} * x^n}{V_{D0} + V_{D1} * x + V_{D2} * x^2 + ... + V_{Dm} * x^m}$$

Possible data types:
Internal: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64
Physical: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64

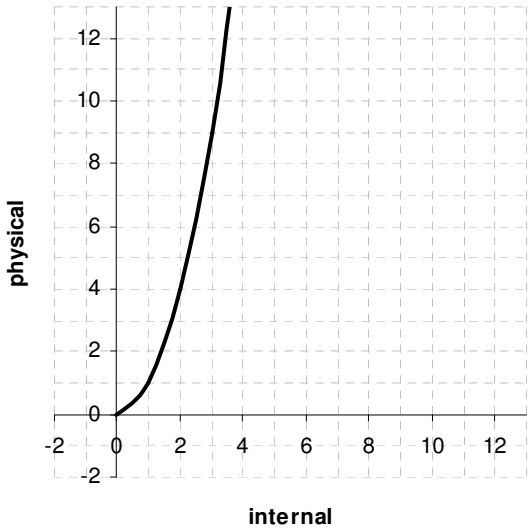


Figure 61 — Rational function

EXAMPLE

$$f(x) = x^2, x \in [0, +\infty)$$

```
<COMPU-METHOD>
      <CATEGORY>RAT-FUNC</CATEGORY>
      <COMPU-INTERNAL-TO-PHYS>
            <COMPU-SCALES>
                  <COMPU-SCALE>
                        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
                        <UPPER-LIMIT INTERVAL-TYPE="INFINITE"/>
                        <COMPU-RATIONAL-COEFFS>
```

```
                                        <COMPU-
NUMERATOR><V>0</V><V>0</V><V>1</V></COMPU-NUMERATOR>
                                        <COMPU-DENOMINATOR><V>1</V></COMPU-DENOMINATOR>
                                </COMPU-RATIONAL-COEFFS>
                        </COMPU-SCALE>
                </COMPU-SCALES>
        </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

7.3.6.6.6 Scale rational function

Like CompuMethods of the SCALE-LINEAR type CompuMethods of the SCALE-RAT-FUNC type can have more than one COMPU-SCALE to provide different calculations for different intervals. The calculation in the reverse direction (physical to internal) is not allowed for this type.

Mathematical function:

$$f(x)=\begin{cases}\dfrac{V_{N0}^1+V_{N1}^1*x+V_{N2}^1*x^2+\ldots+V_{Nn}^1*x^n}{V_{D0}^1+V_{D1}^1*x+V_{D2}^1*x^2+\ldots+V_{Dm}^1*x^m}, x\in I_1 \\ \cdots \\ \dfrac{V_{N0}^k+V_{N1}^k*x+V_{N2}^k*x^2+\ldots+V_{Nn}^k*x^n}{V_{D0}^k+V_{D1}^k*x+V_{D2}^k*x^2+\ldots+V_{Dm}^k*x^m}, x\in I_k\end{cases}$$

where I_1, … , I_k are arbitrary disjunctive intervals.

Possible data types:
Internal: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64
Physical: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64

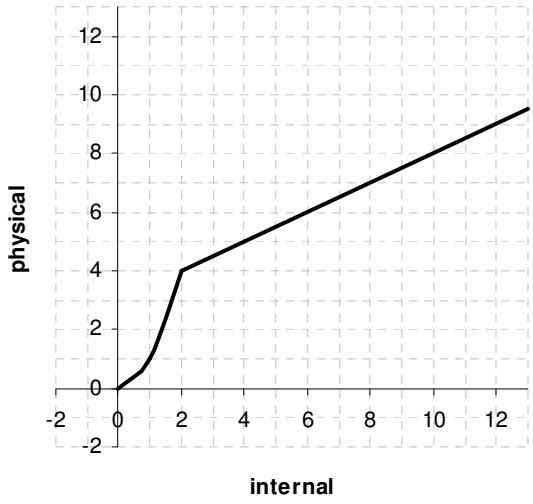


Figure 62 — Scale rational function

EXAMPLE

$$f(x)=\begin{cases}x^2, x\in[0,2) \\ \cdots \\ \dfrac{6+x}{2}, x\in[2,\infty)\end{cases}$$

```
<COMPU-METHOD>
        <CATEGORY>SCALE-RAT-FUNC</CATEGORY>
        <COMPU-INTERNAL-TO-PHYS>
                <COMPU-SCALES>
                        <COMPU-SCALE>
                                <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
                                <UPPER-LIMIT INTERVAL-TYPE="OPEN">2</UPPER-LIMIT>
                                <COMPU-RATIONAL-COEFFS>
                                        <COMPU-NUMERATOR>
                                                <V>0</V>
                                                <V>0</V>
                                                <V>1</V>
                                        </COMPU-NUMERATOR>
```

```
                                <COMPU-DENOMINATOR>
                                        <V>1</V>
                                </COMPU-DENOMINATOR>
                        </COMPU-RATIONAL-COEFFS>
                </COMPU-SCALE>
                <COMPU-SCALE>
                        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
                        <UPPER-LIMIT INTERVAL-TYPE="INFINITE"/>
                        <COMPU-RATIONAL-COEFFS>
                                <COMPU-NUMERATOR>
                                        <V>6</V>
                                        <V>1</V>
                                </COMPU-NUMERATOR>
                                <COMPU-DENOMINATOR>
                                        <V>2</V>
                                </COMPU-DENOMINATOR>
                        </COMPU-RATIONAL-COEFFS>
                </COMPU-SCALE>
        </COMPU-SCALES>
    </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

7.3.6.6.7 Texttable

The type TEXTTABLE is used for transformations of the internal value into a textual expression. COMPU-INTERNAL-TO-PHYS must contain one or more scales. In each scale UPPER-LIMIT and LOWER-LIMIT define an interval. The optional COMPU-DEFAULT-VALUE can be used to define the physical value if the internal value does not lie in any given interval. COMPU-CONST with the data object VT defines the resulting text for the appropriate interval. If the reverse calculation is needed, for each scale the COMPU-INVERSE-VALUE should be used to define the reverse calculation result. In case where LOWER-LIMIT is equal to UPPER-LIMIT, the COMPU-INVERSE-VALUE is optional. If it is omitted the LOWER-LIMIT is used as the inverse value. To guarantee the unambiguous reverse calculation the COMPU-INVERSE-VALUE of all the COMPU-SCALEs with same physical string value must be identical. If the internal data type is a string type the definition of a range is not allowed. In this case LOWER-LIMIT must be identical with UPPER-LIMIT and INTERVAL-TYPEs must be defined as "CLOSED".

Example of a TEXTTABLE function:

internal	physical
[0,5)	"OFF"
[5,9)	"ON"
[9,∞)	"OVERLOAD"

Possible data types:
Internal: A_INT32, A_UINT32, A_UNICODE2STRING, A_BYTEFIELD
Physical: A_UNICODE2STRING

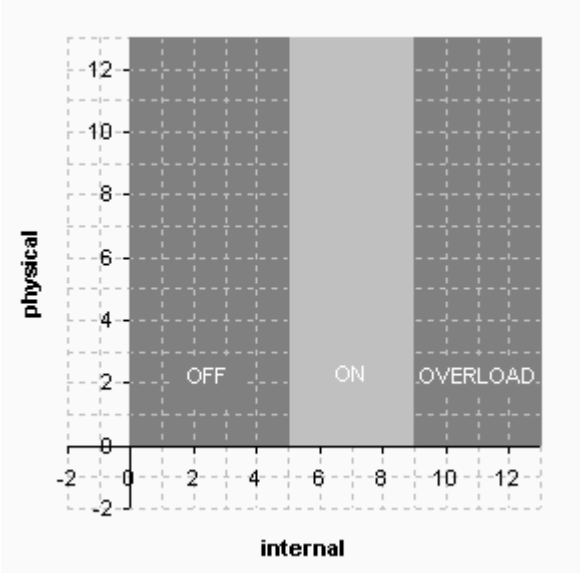


Figure 63 — Texttable

The example shown above is implemented as follows:

```
<COMPU-METHOD>
       <CATEGORY>TEXTTABLE</CATEGORY>
       <COMPU-INTERNAL-TO-PHYS>
              <COMPU-SCALES>
                     <COMPU-SCALE>
                            <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
                            <UPPER-LIMIT INTERVAL-TYPE="OPEN">5</UPPER-LIMIT>
                            <COMPU-INVERSE-VALUE><V>0</V></COMPU-INVERSE-VALUE>
                            <COMPU-CONST><VT>OFF</VT></COMPU-CONST>
                     </COMPU-SCALE>
                     <COMPU-SCALE>
                            <LOWER-LIMIT INTERVAL-TYPE="CLOSED">5</LOWER-LIMIT>
                            <UPPER-LIMIT INTERVAL-TYPE="OPEN">9</UPPER-LIMIT>
                            <COMPU-INVERSE-VALUE><V>5</V></COMPU-INVERSE-VALUE>
                            <COMPU-CONST><VT>ON</VT></COMPU-CONST>
                     </COMPU-SCALE>
                     <COMPU-SCALE>
                            <LOWER-LIMIT INTERVAL-TYPE="CLOSED">9</LOWER-LIMIT>
                            <UPPER-LIMIT INTERVAL-TYPE="INFINITE"/>
                            <COMPU-INVERSE-VALUE><V>9</V></COMPU-INVERSE-VALUE>
                            <COMPU-CONST><VT>OVERLOAD</VT></COMPU-CONST>
                     </COMPU-SCALE>
              </COMPU-SCALES>
              <COMPU-DEFAULT-VALUE><VT>UNDEF</VT></COMPU-DEFAULT-VALUE>
       </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

7.3.6.6.8 Tab interpolated

A CompuMethod of type TAB-INTP defines a set of points. Only LOWER-LIMIT is used for that purpose. The calculation is performed as follows: The internal value is matched against the value intervals in the order they are defined by the COMPU-SCALEs; the limits of a value interval are defined by the LOWER-LIMIT values of two adjacent COMPU-SCALEs. Once the internal value matches a point or an interval the iteration is to be stopped and the physical value is to be calculated via linear interpolation. If the internal value is less than the smallest defined X-value or greater than the greatest defined X-value a runtime error must be indicated by the runtime system. The calculation in the reverse direction (physical to internal) is performed in an analogous manner and also in the same order, with the difference that the physical value is tested against the value intervals, which are defined by the COMPU-CONST values of two adjacent COMPU-SCALEs. The first COMPU-SCALE matching the value will be used to calculate it in reverse direction.

Example of a TAB-INTP function:

Internal	Physical
0	0
2	5
4	8
8	10
10	10

Possible data types:
Internal: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64
Physical: A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64

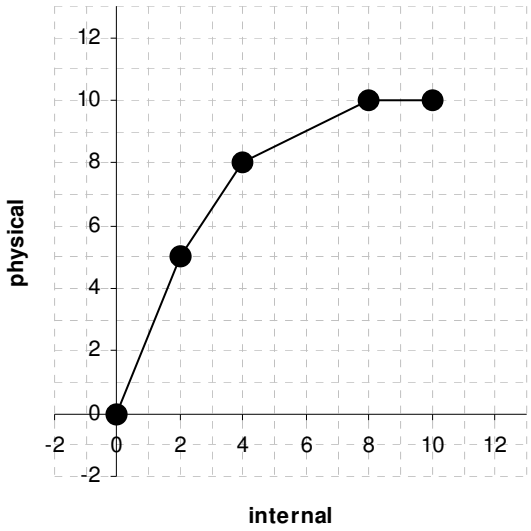


Figure 64 — Tab interpolated

The following XML code shows the implementation of this example:

```
<COMPU-METHOD>
	<CATEGORY>TAB-INTP</CATEGORY>
	<COMPU-INTERNAL-TO-PHYS>
		<COMPU-SCALES>
			<COMPU-SCALE>
				<LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
				<COMPU-CONST><V>0</V></COMPU-CONST>
			</COMPU-SCALE>
			<COMPU-SCALE>
				<LOWER-LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
				<COMPU-CONST><V>5</V></COMPU-CONST>
			</COMPU-SCALE>
			<COMPU-SCALE>
				<LOWER-LIMIT INTERVAL-TYPE="CLOSED">4</LOWER-LIMIT>
				<COMPU-CONST><V>8</V></COMPU-CONST>
			</COMPU-SCALE>
			<COMPU-SCALE>
				<LOWER-LIMIT INTERVAL-TYPE="CLOSED">8</LOWER-LIMIT>
				<COMPU-CONST><V>10</V></COMPU-CONST>
			</COMPU-SCALE>
			<COMPU-SCALE>
				<LOWER-LIMIT INTERVAL-TYPE="CLOSED">10</LOWER-LIMIT>
				<COMPU-CONST><V>10</V></COMPU-CONST>
			</COMPU-SCALE>
		</COMPU-SCALES>
	</COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

7.3.6.6.9 Computational code

A CompuMethod of type COMPUCODE is used when a computer program should do the computation. This must be coded in Java programming language. Java libraries allowed to use are listed in Table 9. A java class with computational code has to implement the interface `I_CompuCode`.
Interface of computational code:

```
public interface I_CompuCode {
    public Object compute(Object input);
```

```
}
```

The input parameter and return value can be one of the following types.

- java.lang.String
- java.lang.Byte[] or byte[]
- java.lang.Integer
- java.lang.BigInteger
- java.lang.Float
- java.lang.Double

Table 9 — Java packages for computational code

Package	Description
Java.lang	Provides classes that are fundamental to the design of the Java programming language.
Java.math	Provides classes for performing arbitrary-precision integer arithmetic (BigInteger) and arbitrary-precision decimal arithmetic (BigDecimal).
Java.text	Provides Classes and interfaces for handling text, dates, numbers, and messages in a manner independent of natural languages.
Java.util	Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalisation, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).

No instance of COMPU-SCALE is allowed for this type of computational method. All data types for physical and internal value are applicable. The structure COMPU-METHOD provides the possibility to reference java code for both directions of computation: COMPU-INTERNAL-TO-PHYS for computation from internal type to physical and COMPU-PHYS-TO-INTERNAL for physical to internal.
COMPU-PHYS-TO-INTERNAL can only be used for this kind of CompuMethod.
To refer to the right computational code the name of the source file or class file has to be placed as value of the member CODE-FILE. ENCRYPTION provides a possibility to put in the information about encryption method if used. This member defines all details necessary for the encryption. The definition takes place in a user specific way and is not specified in this document. There are three storage kinds for computational java code which are defined by the member SYNTAX: “JAVA” for java source code, “CLASS” for java byte code (compiled) and “JAR” for java archive. REVISION indicates the version of the file to be used. It can be empty if no versioning is supported. When storing as java archive an ENTRYPOINT specifies the name of the class containing the program code. The standard only supports jobs written in Java. If jobs in another syntax (e.g. a Windows DLL) need to be supported, a corresponding value for the SYNTAX field and the entrypoint into the DLL can be included.
EXAMPLE A COMPUCODE function defined in XML referencing a java file:

```
<COMPU-METHOD>
       <CATEGORY>COMPUCODE</CATEGORY>
       <COMPU-INTERNAL-TO-PHYS>
              <PROG-CODE>
                     <CODE-FILE>myCompucode2.java</CODE-FILE>
                     <SYNTAX>JAVA</SYNTAX>
                     <REVISION>1.0</REVISION>
              </PROG-CODE>
       </COMPU-INTERNAL-TO-PHYS>
       <COMPU-PHYS-TO-INTERNAL>
```

```
            <PROG-CODE>
                  <CODE-FILE>myCompucode2.java</CODE-FILE>
                  <SYNTAX>JAVA</SYNTAX>
                  <REVISION>1.0</REVISION>
            </PROG-CODE>
      </COMPU-PHYS-TO-INTERNAL>
</COMPU-METHOD>
```

EXAMPLE A computational code in a java file (e.g. my_compucode1.java):

```
import java.lang.* ;
public class myCompucode1
implements I_CompuCode {
    public Object compute(Object input){
        .. .. ..
    }
}
```

EXAMPLE A COMPUCODE function defined in XML referencing a java class

```
<COMPU-METHOD>
      <CATEGORY>COMPUCODE</CATEGORY>
      <COMPU-INTERNAL-TO-PHYS>
            <PROG-CODE>
                  <CODE-FILE>myCompucode1</CODE-FILE>
                  <SYNTAX>CLASS</SYNTAX>
                  <REVISION>1.0</REVISION>
            </PROG-CODE>
      </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

EXAMPLE Referencing the class "myCompucode1" in the archive "allCompucode.jar"

```
<COMPU-METHOD>
      <CATEGORY>COMPUCODE</CATEGORY>
      <COMPU-INTERNAL-TO-PHYS>
            <PROG-CODE>
                  <CODE-FILE>allCompucode.jar</CODE-FILE>
                  <SYNTAX>JAR</SYNTAX>
                  <REVISION>1.0</REVISION>
                  <ENTRYPOINT>myCompucode1</ENTRYPOINT>
            </PROG-CODE>
      </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

7.3.6.7 UNITS

Usually units are used to augment the value with additional information like "m/s" or "liter". That is necessary for a correct interpretation of the physical value for input and output processes. The application shows the DISPLAY-NAME of the unit besides the physical value of the diagnostic data.
The conversion of values into other units like "km/h" into "miles/h" is also possible. Therefore the unit involves information about its physical dimensions. The substructure of physical dimensions defines all used quantities in the SI-System (e.g. velocity as length/time corresponds to m/s). The unit references one physical dimension. If the physical dimension of two units is identical a conversion between them is possible.

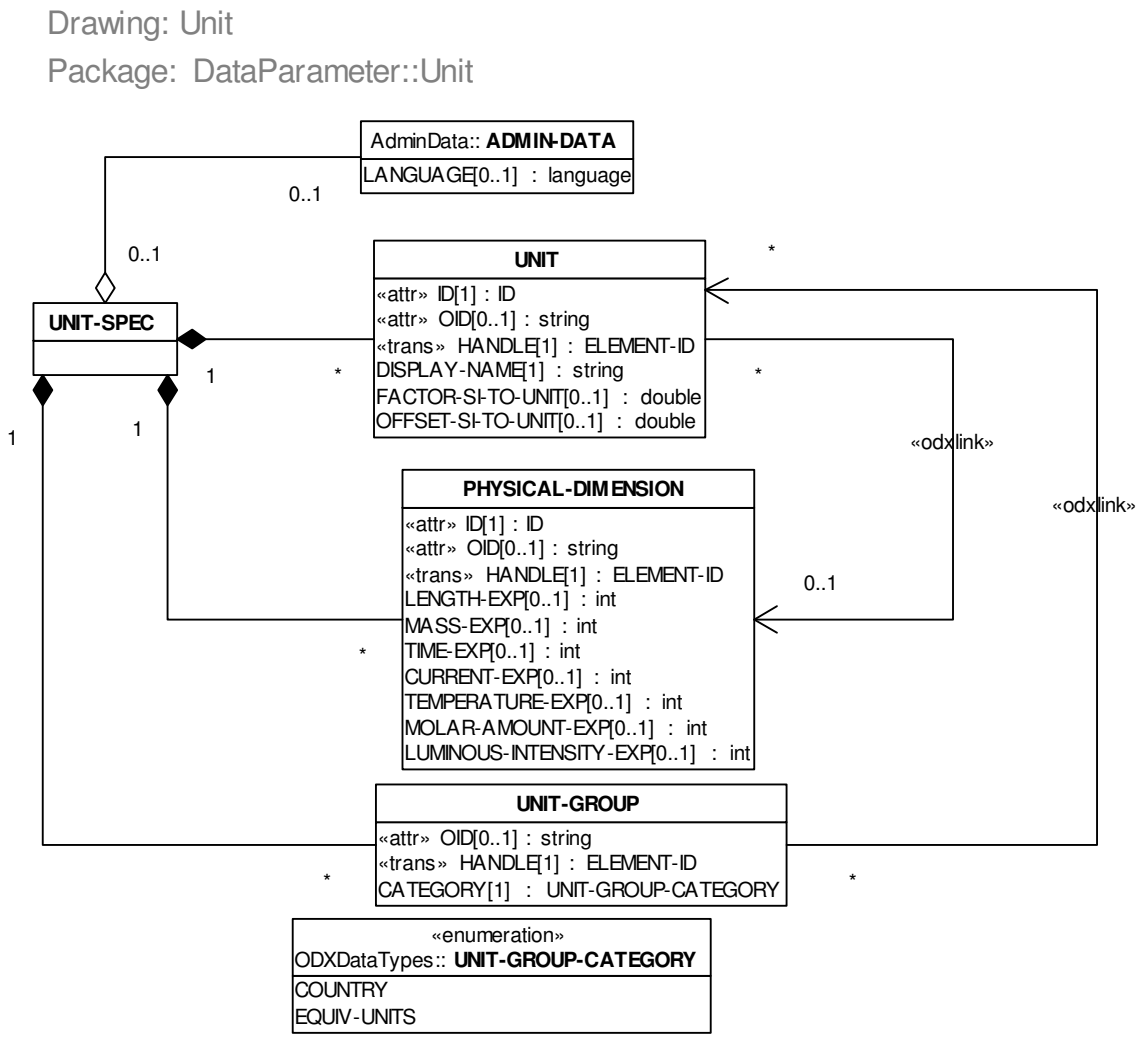


Figure 65 — Unit

In the model implemented, all units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined. For the calculation from SI-unit to the defined unit the factor (FACTOR-SI-TO-UNIT) and the offset (OFFSET-SI-TO-UNIT) are applied:

UNIT = SI-UNIT * FACTOR-SI-TO-UNIT + OFFSET-SI-TO-UNIT

For the calculation from a unit to SI-unit the reciprocal of the factor (FACTOR-SI-TO-UNIT) and the negation of the offset (OFFSET-SI-TO-UNIT) are applied:

SI-UNIT = (UNIT - OFFSET-SI-TO-UNIT) / FACTOR-SI-TO-UNIT

There is also a possibility to define units without referencing a PHYSICAL-DIMENSION. In this case, FACTOR-SI-TO-UNIT and OFFSET-SI-TO-UNIT must not be specified. Only the DISPAY-NAME has to be given which is used for displaying. It is recommended to avoid the definition of such units, though.

In the following example you can see some definitions of units and the corresponding conversion from one unit to another.

EXAMPLE

```
<UNIT-SPEC>
       <UNITS>
              <UNIT ID="Unit_Celsius">
                     <SHORT-NAME>Unit_Celsius</SHORT-NAME>
                     <DISPLAY-NAME>°C</DISPLAY-NAME>
              </UNIT>
              <UNIT ID="Unit_Kmh">
                     <SHORT-NAME>Unit_Kmh</SHORT-NAME>
                     <DISPLAY-NAME>km/h</DISPLAY-NAME>
                     <FACTOR-SI-TO-UNIT>3.6</FACTOR-SI-TO-UNIT>
                     <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-UNIT>
                     <PHYSICAL-DIMENSION-REF ID-REF="PD_Velocity"/>
              </UNIT>
              <UNIT ID="Unit_MilesPerHour">
                     <SHORT-NAME>Unit_MilesPerHour</SHORT-NAME>
                     <DISPLAY-NAME>mph</DISPLAY-NAME>
                     <FACTOR-SI-TO-UNIT>2.23741</FACTOR-SI-TO-UNIT>
                     <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-UNIT>
                     <PHYSICAL-DIMENSION-REF ID-REF="PD_Velocity"/>
              </UNIT>
       </UNITS>
       <PHYSICAL-DIMENSIONS>
              <PHYSICAL-DIMENSION ID="PD_Temp">
                     <SHORT-NAME>PD_Temp</SHORT-NAME>
                     <TEMPERATURE-EXP>1</TEMPERATURE-EXP>
              </PHYSICAL-DIMENSION>
              <PHYSICAL-DIMENSION ID="PD_Velocity">
                     <SHORT-NAME>PD_Velocity</SHORT-NAME>
                     <LONG-NAME>Meter Per Second</LONG-NAME>
                     <LENGTH-EXP>1</LENGTH-EXP>
                     <TIME-EXP>-1</TIME-EXP>
              </PHYSICAL-DIMENSION>
       </PHYSICAL-DIMENSIONS>
</UNIT-SPEC>
```

For the conversion from unit "km/h" in "mph" via SI-UNIT (m/s) the following equations result:

Step1: SI-UNIT = (km/h – 0) / 3.6

Step2: mph = SI-UNIT * 2.23741 + 0

There is also a unit "Unit_Celsius" in this example, which does not reference a PHYSICAL-DIMENSION. If a DATA-OBJECT-PROP refers to this unit, the physical value, extracted using this DOP, is simply displayed with the string "°C" behind it.

Unit groups

Units can be grouped with the help of UNIT-GROUP. Each UNIT-GROUP may refer to an arbitrary number of UNITs. The member CATEGORY of the UNIT-GROUP denotes the unit system that the UNIT-GROUP's units are associated to. In this way, country-specific unit systems (CATEGORY="COUNTRY") can be defined as well as specific unit systems for certain application domains. For example, we can define a group of equivalent units, which are used in different countries, by setting CATEGORY="EQUIV-UNITS". For example, KmPerHour and MilesPerHour could be combined to one group of this category named "vehicle_speed". The unit MeterPerSec would not belong to this group because it is normally not used for vehicle speed. But all of the mentioned units could be combined to one group named "speed".

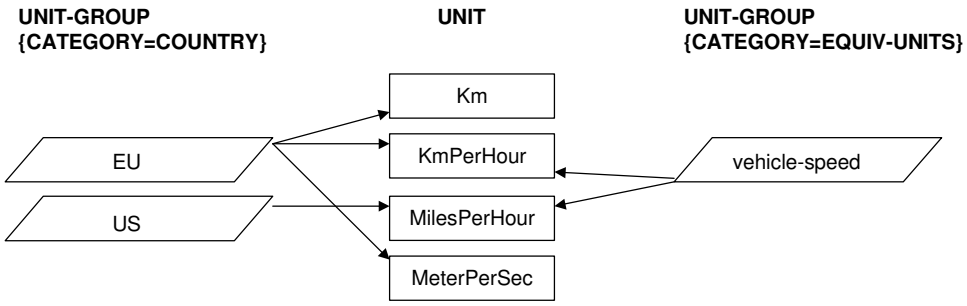


Figure 66 — UNIT-GROUP example

An ODX tool can use information stored in UNIT-GROUPS, for example, to present a list of the available unit categories to the user; then, the user can select the appropriate unit system from this list and the following computations would be performed using the units belonging to this system.
A UNIT-GROUP always refers to UNITs that are defined in the same UNIT-SPEC as the UNIT-GROUP itself.
EXAMPLE

```
<UNIT-SPEC>
      <UNIT-GROUPS>
            <UNIT-GROUP>
                  <SHORT-NAME>europe_unitgroup</SHORT-NAME>
                  <CATEGORY>COUNTRY</CATEGORY>
                  <UNIT-REFS>
                        <UNIT-REF ID-REF="Unit_Meter"/>
                        <UNIT-REF ID-REF="Unit_Kmh"/>
                  </UNIT-REFS>
            </UNIT-GROUP>
            <UNIT-GROUP>
                  <SHORT-NAME>usa_unitgroup</SHORT-NAME>
                  <CATEGORY>COUNTRY</CATEGORY>
                  <UNIT-REFS>
                        <UNIT-REF ID-REF="Unit_Foot"/>
                        <UNIT-REF ID-REF="Unit_Mph"/>
                  </UNIT-REFS>
            </UNIT-GROUP>
            <UNIT-GROUP>
                  <SHORT-NAME>VehicleSpeed</SHORT-NAME>
                  <LONG-NAME>Vehicle Speed</LONG-NAME>
                  <CATEGORY>EQUIV-UNITS</CATEGORY>
                  <UNIT-REFS>
                        <UNIT-REF ID-REF="Unit_Kmh"/>
                        <UNIT-REF ID-REF="Unit_Mph"/>
                  </UNIT-REFS>
            </UNIT-GROUP>
      </UNIT-GROUPS>
</UNIT-SPEC>
```

7.3.6.8 DTC DATA OBJECT PROPERTY

The interpretation of DTCs (diagnostic trouble codes) received from the ECU is a core functional task during diagnostics. The runtime system usually offers the detailed information for each received DTC. Additionally, environment data can also be given for a DTC. As a consequence, DTC-DOPs (a special variant of a DOP) and ENV-DATA-DESCs (description of environment data records) are introduced for the handling of DTCs.

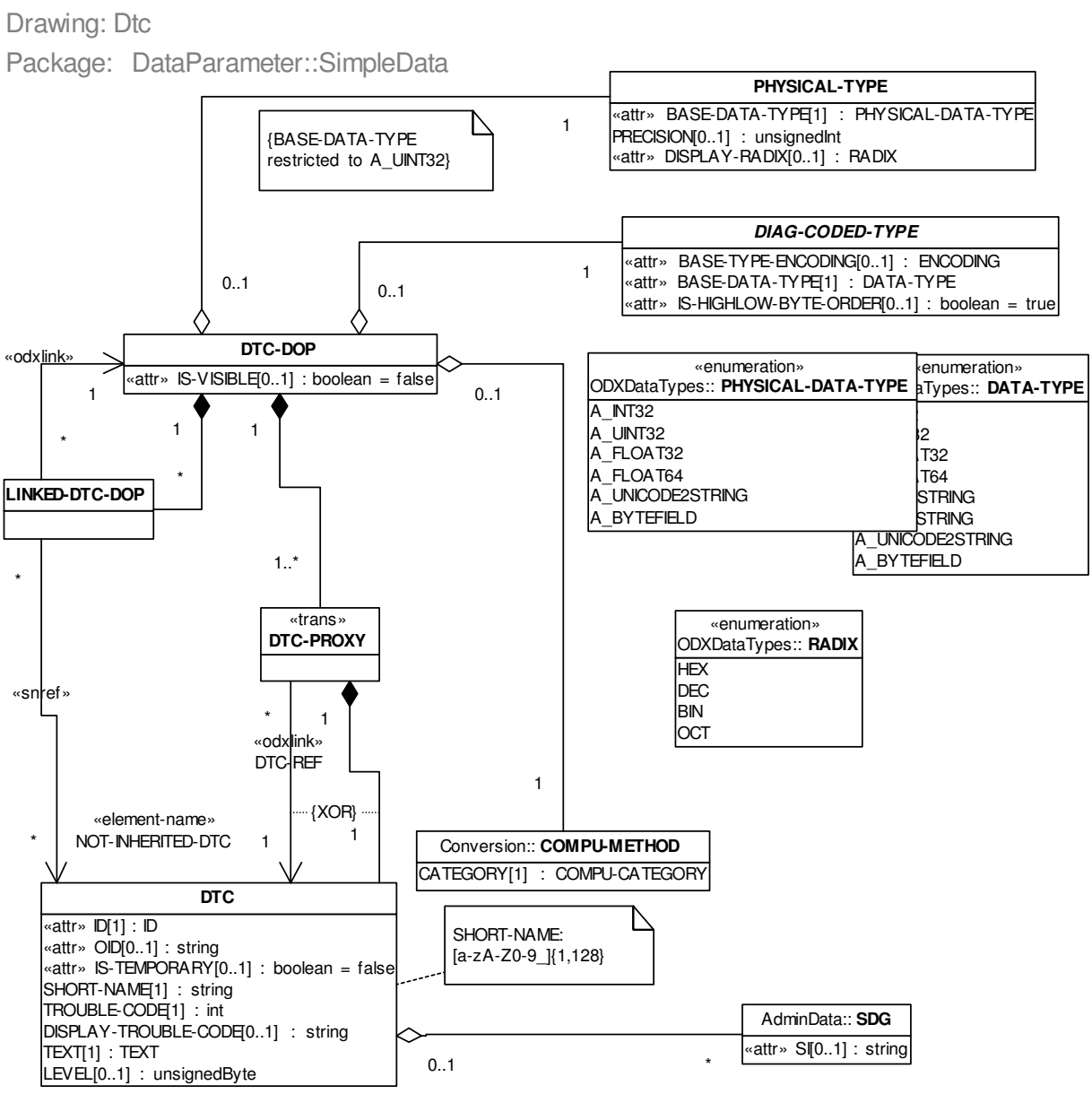


Figure 67 — DTC

A DTC-DOP is an extended DOP (data object property) with data extraction parameters (DIAG-CODED-TYPE), a conversion method (COMPU-METHOD), and a result type (PHYSICAL-TYPE). Additionally, it provides a set of DTC objects, which contain the TROUBLE-CODE itself, a TEXT with an error message for this DTC and a LEVEL.
In an early stage of the ODX data a DTC may be marked with the attribute IS-TEMPORARY. These temporary DTCs only serve for documentation purposes during the coordination between manufacturer and supplier. They should exist only during the design phase of the ODX data. An MCD 3D / MVCI server should ignore all temporary DTCs.
The value of TROUBLE-CODE is an integer number (A_UINT32) that is used for matching (see "Processing of DTC-DOP" below). As a consequence, the BASE-DATA-TYPE member of PHYSICAL-TYPE must be set to A_UINT32.

The LEVEL is used to filter the DTCs. Its value domain is 1...255. The lower the number, the higher the DTC level. The runtime system can provide a method to request DTCs of a given level range. The method has two parameters for the closed interval range of the DTC levels to be returned. E.g. [10...200] returns all DTCs with levels set to numbers between 10 and 200 inclusively. The default value for a DTC level is 1.
A DISPLAY-TROUBLE-CODE (for example the SAE code) can be used to specify the display value for this DTC if it differs from the numeric TROUBLE-CODE. DISPLAY-RADIX is evaluated if the corresponding DTC does not have the sub-element DISPLAY-TROUBLE-CODE and indicates then the way the TROUBLE-CODE must be displayed. Specific information like error set conditions may be added as SDGs (see section 7.1.2.1 for a more detailed description of SDGs).
To provide the reuse of DTC objects, DTC-DOP may reference to DTCs that are contained in other diagnostic layers (see explanation below).
For better understanding, the DTC-DOP can be visualized as a global table of error codes, which may look like this:

Table 10 — Table visualisation of DTC-DOP data

SHORT-NAME	TROUBLE-CODE	DISPLAY-TROUBLE-CODE	TEXT	LEVEL (1-255)
DTC_0120	288		Throttle Position Malf.	5
DTC_0130	304	B0130	Sensor Circuit Malf.	1
...				

The following figure shows the structure of the service ReadDiagnosticTroubleCodesByStatus (service id = 18h) of ISO 14230. The service consists of a request and a response. Each of them consists of several parameters on its part. The first parameter of the response is the service id + 40h, the second gives the number of DTCs. The parameter "ListOfDTCAndStatus" refers to an END-OF-PDU-FIELD, which indicates, that the referenced STRUCTURE is repeated until the end of the PDU. This structure contains two parameters. One of them refers to a DTC-DOP.

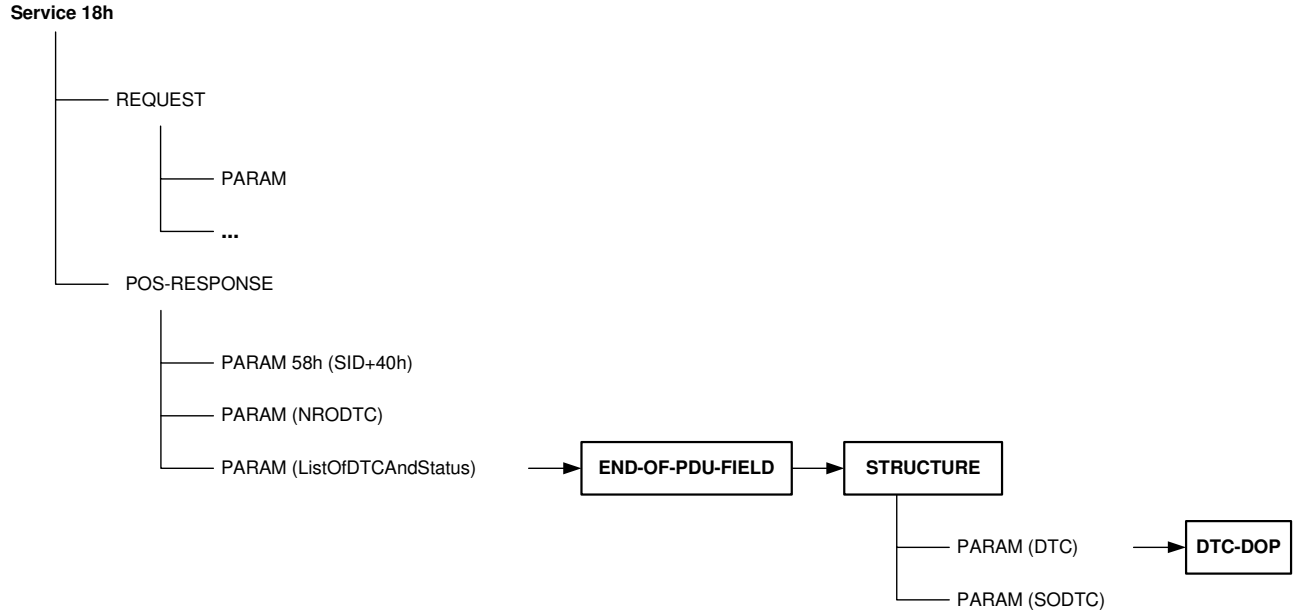


Figure 68 — Structure of Service ReadDiagnosticTroubleCodesByStatus (18h) of ISO 14230

If a response parameter refers to a DTC-DOP, the runtime system has to extract the value of DTC in the same manner as other response parameters using DIAG-CODED-TYPE, PHYSICAL-TYPE and COMPU-METHOD. The BASE-DATA-TYPE of PHYSICAL- and DIAG-CODED-TYPE is restricted to A_UINT32. The physical DTC value, i.e. the result of the execution of the COMPU-METHOD of the DTC-DOP is then used as a switch-key (Figure 69, switch-key = 0120h). Afterwards, it is taken to find the matching DTC object by looking for a matching TROUBLE-CODE. A DTC object with TROUBLE-CODE = switch-key is finally displayed to the user.

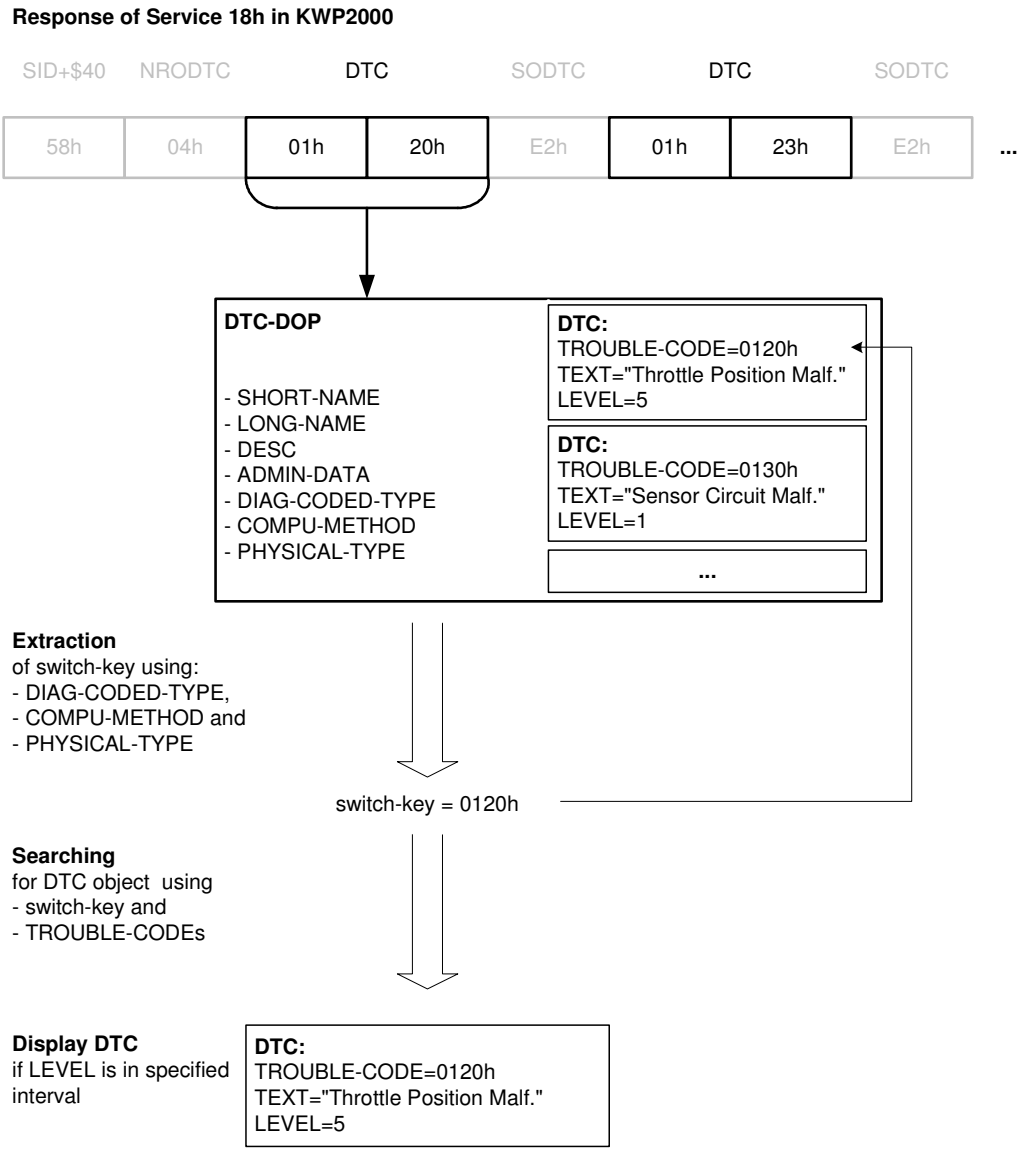


Figure 69 — Processing of DTC-DOP

EXAMPLE DTC-DOP in XML

```
<DTC-DOP ID="DTCDOP_All">
       <SHORT-NAME>DTCDOP_All</SHORT-NAME>
       <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-
TYPE">
              <BIT-LENGTH>16</BIT-LENGTH>
       </DIAG-CODED-TYPE>
       <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
       <COMPU-METHOD>
              <CATEGORY>IDENTICAL</CATEGORY>
```

```
        </COMPU-METHOD>
        <DTCS>
                <DTC ID="DTC0120">
                        <SHORT-NAME>DTC0120</SHORT-NAME>
                        <TROUBLE-CODE>288</TROUBLE-CODE>
                        <TEXT>Throttle Position Malf.</TEXT>
                        <LEVEL>5</LEVEL>
                </DTC>
                <DTC ID="DTC0130">
                        <SHORT-NAME>DTC0130</SHORT-NAME>
                        <TROUBLE-CODE>304</TROUBLE-CODE>
                        <DISPLAY-TROUBLE-CODE>B0130</DISPLAY-TROUBLE-CODE>
                        <TEXT>Sensor Circuit Malf.</TEXT>
                        <LEVEL>1</LEVEL>
                </DTC>
        </DTCS>
</DTC-DOP>
```

DTC-DOPs can be reused by means of inheritance or by referencing the objects in a ECU-SHARED-DATA layer. The DTC objects can also be reused. For example, if in a lower layer new DTCs should be added to the existing DTC list defined in the upper layer (DTC-DOP A in the figure below), a new object has to be created in the lower layer (DTC-DOP B). This object references to the DTCs to be reused from the object A and defines the new DTCs that should be added. The advantage of this approach is that if certain DTCs are not to be used in the lower layer, they can be eliminated in an easy manner: not used DTCs are simply not referenced.

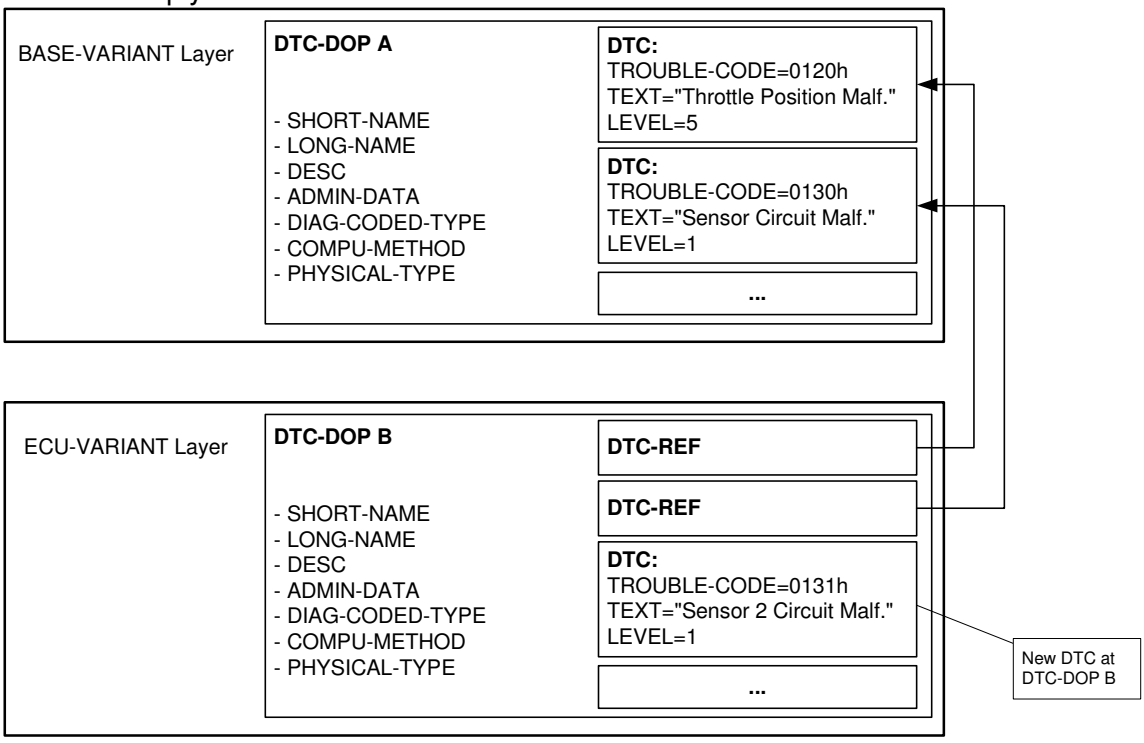


Figure 70 — Reuse of DTCs

A DTC-DOP may also be used in a REQUEST. A request parameter referencing a DTC-DOP provides the possibility to enter an existing DTC by TROUBLE-CODE. The COMPU-METHOD of the DTC-DOP is used to calculate the internal value which has to be passed to the ECU.
A further alternative to reuse DTC-DOPs and their containing DTCs is to work with LINKED-DTC-DOPs. This feature allows the use of categorised DTCs inside one DTC-

DOP. That means a DTC-DOP might be defined which contains DTCs, describing for instance common behaviour and also a DTC-DOP might be defined which contains DTCs, describing for instance ECU specific behaviour in particular. This helps to prevent redundancy.
To cite an example the ODX-container for the base variant of an ECU “Engine Control Module” shall list all the DTCs, applicable for this ECU. This list of necessary DTCs is made up of common DTCs (that might be important for all ECUs) and DTCs only applicable for this ECU. In this example the DTC-DOP defined in the base variant contains all ECU specific DTCs. In an ECU-SHARED-DATA layer a DTC-DOP is defined that contains all common DTCs. To get those common DTCs in focus inside the DTC-DOP with the ECU applicable DTCs in the base variant of the “Engine Control Module” ECU a LINKED-DTC-DOP reference has to be established within the DTC-DOP defining the ECU specific DTCs. Now all the common DTCs referenced via the LINKED-DTC-DOP are also visible inside the DTC-DOP with the ECU specific DTCs.
The most important circumstance with LINKED-DTC-DOPs is, that all listed DTCs within such a linked DTC-DOP are known in the layer, where the LINKED-DTC-DOP is established. The DTCs don’t need to be referenced via a DTC-REF to use them.
The below mentioned XML code example will show the described issue. The listed ECU-SHARED-DATA container shows the DTC-DOP with the common DTCs.

```
<ECU-SHARED-DATA ID="ES_DataPool">
       <SHORT-NAME>ES_DataPool</SHORT-NAME>
       <LONG-NAME>Data Pool</LONG-NAME>
       <DIAG-DATA-DICTIONARY-SPEC>
              <DTC-DOPS>
                     <DTC-DOP ID="DTCDOP_CommonDTCs">
                            <SHORT-NAME>DTCDOP_CommonDTCs</SHORT-NAME>
                            <LONG-NAME>Common DTCs</LONG-NAME>
                            <DESC>
                                   <p>The following described DTCs are applicable
for all ECUs</p>
                            </DESC>
                            <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"
xsi:type="STANDARD-LENGTH-TYPE">
                                   <BIT-LENGTH>16</BIT-LENGTH>
                            </DIAG-CODED-TYPE>
                            <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
                            <COMPU-METHOD>
                                   <CATEGORY>IDENTICAL</CATEGORY>
                            </COMPU-METHOD>
                            <DTCS>
                                   <DTC ID="DTC3004">
                                          <SHORT-NAME>DTC3004</SHORT-NAME>
                                          <TROUBLE-CODE>3004</TROUBLE-CODE>
                                          <DISPLAY-TROUBLE-CODE>P0BBC</DISPLAY-
TROUBLE-CODE>
                                          <TEXT>Fault path for ACC Sensor
Error</TEXT>
                                          <LEVEL>2</LEVEL>
                                   </DTC>
                                   <DTC ID="DTC0130">
                                          <SHORT-NAME>DTC0130</SHORT-NAME>
                                          <TROUBLE-CODE>304</TROUBLE-CODE>
                                          <DISPLAY-TROUBLE-CODE>B0130</DISPLAY-
TROUBLE-CODE>
                                          <TEXT>Sensor Circuit Malf.</TEXT>
                                          <LEVEL>1</LEVEL>
                                   </DTC>
                            </DTCS>
                     </DTC-DOP>
              </DTC-DOPS>
       </DIAG-DATA-DICTIONARY-SPEC>
</ECU-SHARED-DATA>
```

The listed BASE-VARIANT container shows the DTC-DOP with the ECU specific DTCs and also shows the reference (via LINKED-DTC-DOP) to the common DTCs that are also important for this ECU.

```
<BASE-VARIANT ID="BV_EngineControlModule">
	<SHORT-NAME>BV_EngineControlModule</SHORT-NAME>
	<LONG-NAME>Engine control module</LONG-NAME>
	<DIAG-DATA-DICTIONARY-SPEC>
		<DTC-DOPS>
			<DTC-DOP ID="DTC_DOP_ListOfDTCsForEngineControlModule">
				<SHORT-
NAME>DTC_DOP_ListOfDTCsForEngineControlModule</SHORT-NAME>
				<LONG-NAME>List of DTCs for engine control
module</LONG-NAME>
				<DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"
xsi:type="STANDARD-LENGTH-TYPE">
					<BIT-LENGTH>16</BIT-LENGTH>
				</DIAG-CODED-TYPE>
				<PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
				<COMPU-METHOD>
					<CATEGORY>IDENTICAL</CATEGORY>
				</COMPU-METHOD>
				<DTCS>
					<DTC ID="DTC_0120">
						<SHORT-NAME>DTC0120</SHORT-NAME>
						<TROUBLE-CODE>288</TROUBLE-CODE>
						<DISPLAY-TROUBLE-CODE>B0120</DISPLAY-
TROUBLE-CODE>
						<TEXT>Throttle Position Malf.</TEXT>
						<LEVEL>5</LEVEL>
					</DTC>
				</DTCS>
				<LINKED-DTC-DOPS>
					<LINKED-DTC-DOP>
						<DTC-DOP-REF ID-REF="DTCDOP_CommonDTCs"
DOCREF="ES_DataPool" DOCTYPE="LAYER"/>
					</LINKED-DTC-DOP>
				</LINKED-DTC-DOPS>
			</DTC-DOP>
		</DTC-DOPS>
	</DIAG-DATA-DICTIONARY-SPEC>
</BASE-VARIANT>
```

To make it clear, an application now shows the two DTCs DTC3004, DTC0130 and the DTC0120.
It also could be possible that not all DTCs, made available with the DTC-DOP for the common DTCs via LINKED-DTC-DOP, are necessary for the ECU “Engine Control Module”. Therefore the NOT-INHERITED-DTC-SNREF element can be used to exclude the unwanted DTCs from the common DTC data pool.

```
<LINKED-DTC-DOP>
	<NOT-INHERITED-DTC-SNREFS>
		<NOT-INHERITED-DTC-SNREF SHORT-NAME="DTC3004"/>
	</NOT-INHERITED-DTC-SNREFS>
	<DTC-DOP-REF ID-REF="DTCDOP_CommonDTCs" DOCREF="ES_DataPool"
DOCTYPE="LAYER"/>
</LINKED-DTC-DOP>
```

An application now shows the DTCs DTC0130 and the DTC0120 as the possible DTCs.
A DTC-DOP may also be used in REQUESTs (see Table 5). Services $17 in KWP and Service $1906, $1910 are examples for services containing DTCs in a request. A VALUE

referencing a DTC-DOP can be used to describe this in ODX. The 3D-System offers the possibility to enter the TROUBLE-CODE or the DISPLAY-TROUBLE-CODE. The COMPU-METHOD of the DTC-DOP is used to calculate the coded value which has to be passed to the ECU in this case.

7.3.6.9 ENVIRONMENT DATA DESCRIPTION

A DTC may be followed by environment data that describes the circumstances in which the error occurred (see service ReadStatusOfDTC (service id = 17h) of ISO 14230). ENV-DATA-DESC is a complex DOP that is used to define the interpretation of environment data. Since environment data may be defined individually for each DTC, a reference to the response parameter PARAM-SNREF is used to indicate a switch-key e.g. the DTC in the PDU. This parameter must own a simple DOP, a DIAG-CODED-TYPE and a PHYSICAL-TYPE in order to extract the switch-key value. Several ENV-DATA objects are used to describe the structure of the environment data. An ENV-DATA object is selected for interpretation of environment data if one of its DTC-VALUEs matches the current value of the switch-key. More than one DTC-VALUE may be given for one ENV-DATA if the corresponding DTCs have the same structure of environment data. The values have to be given in their physical representation.
The BIT-POSITION of a PARAM using an ENV-DATA directly or via any kind of COMPLEX-DOP must be equal to "0" because an ENV-DATA is a BASIC-STRUCTURE that starts always at byte edge.
In many cases a set of environment data is valid for all DTCs. To avoid duplicate specification of these common environmental data as PARAMs within all affected ENV-DATA objects, a separate ENV-DATA with a special flag ALL-VALUE is intended. The ALL-VALUE ENVDATA is expected in every ENVDATA message stream and is evaluated first. After that the specific DTC-VALUE's ENVDATA is expected in the message stream.
To provide the reuse of ENV-DATA objects, ENV-DATA-DESC may reference ENV-DATAs that are contained in other diagnostic layers. See also Figure 76 and Figure 77. Usage of ENV-DATA-DESC in a FIELD means that the ENV-DATA is repeated in a response.

Drawing: EnvDataDesc
Package: ComplexData

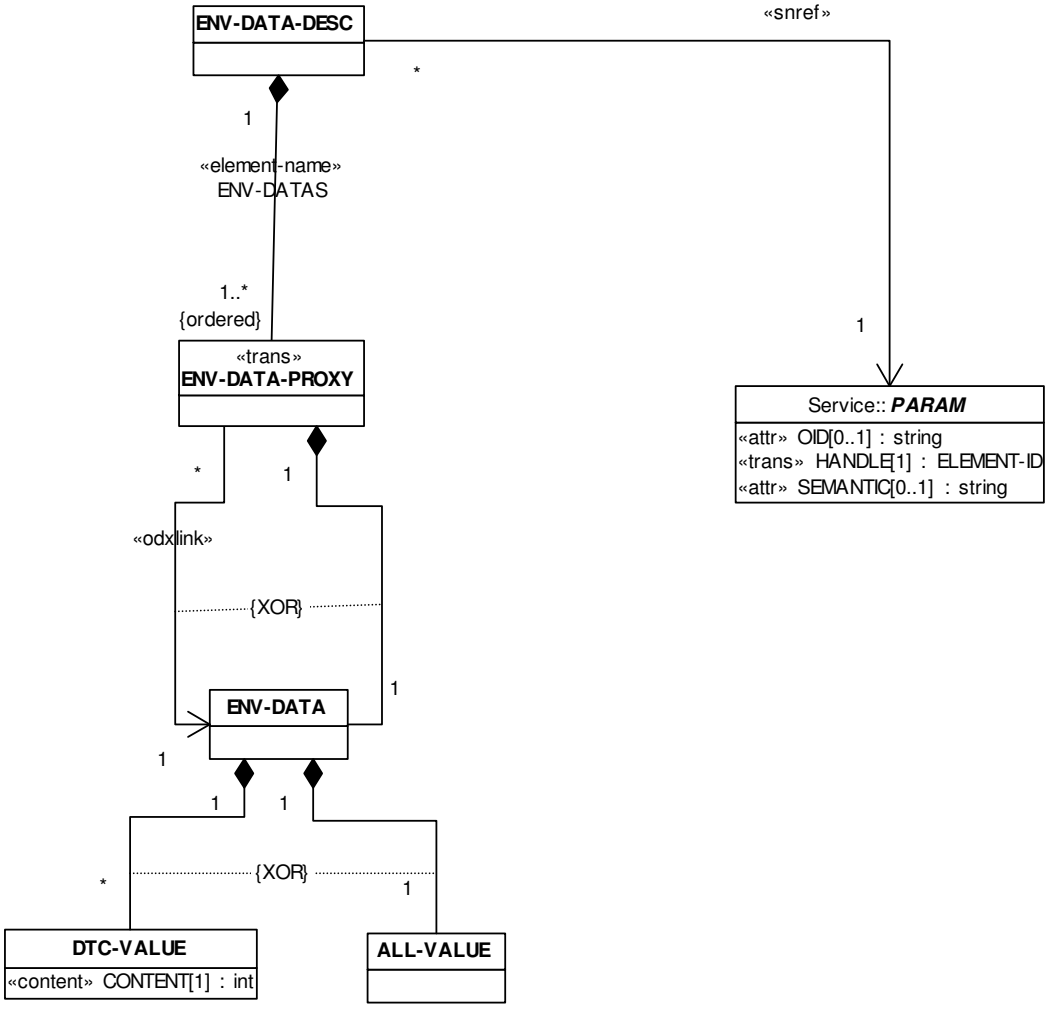


Figure 71 — Structure of ENV-DATA-DESC

Service ReadStatusOfDTC (service id = 17h) is used in ISO 14230 to access the environment data. The first parameter of the response is the serviceId + 40h, the second gives the number of DTCs. The following parameters are the DTC, the status of DTC (SODTC) and the environment data.

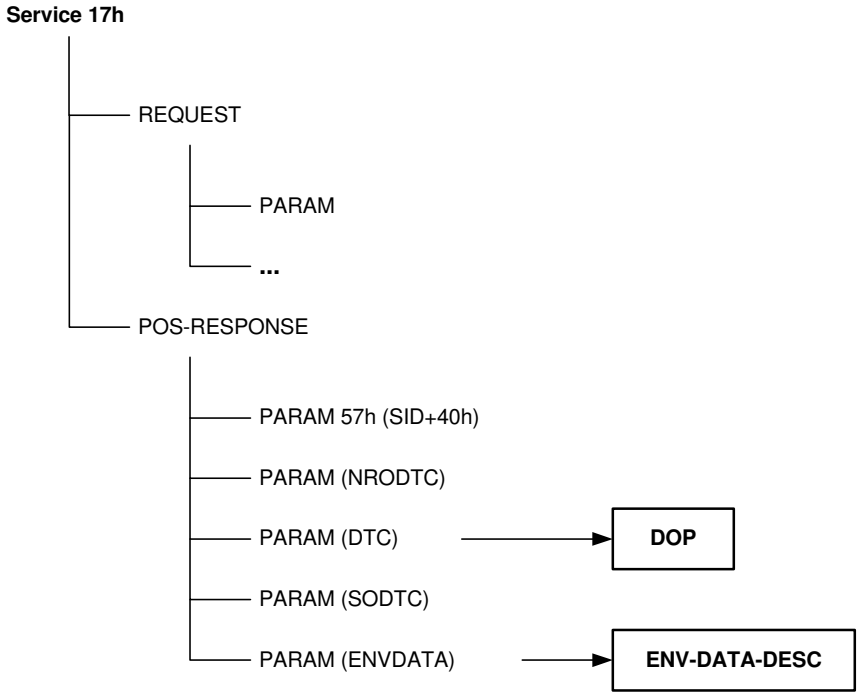


Figure 72 — Structure of Service 17h in ISO 14230

If the runtime system detects a reference to ENV-DATA-DESC during processing the response, it must determine the physical value from the response parameter referenced by PARAM-SNREF. This value is used as a switch-key and must be of type A_UINT32 (PHYSICAL-TYPE.BASE-DATA-TYPE of the referenced DOP must be equal to "A_UINT32"). Then the runtime system has to detect whether an ENV-DATA object with the flag ALL-VALUE exists. Afterwards, the runtime system searches for all ENV-DATA objects that contain the switch-key as a VALUE.
For each ENV-DATA found, all contained PARAMs are processed in order of their BYTE- and BIT-POSITION. These positions are given relatively to the start position of the ENV-DATA-DESC in the PDU. If several PARAMs share the same BYTE- and BIT-POSITION then they are ordered according to the position in the ODX instance. This implies that the order of PARAMs within an ENV-DATA and of ENV-DATAs within an ENV-DATA-DESC must not be changed by the application. In this way the author of the ODX document can define the order, which is later used to sort the data items for displaying. If no ENV-DATA with ALL-VALUE flag and no ENV-DATA with DTC-VALUE = switch-key are found, the runtime system has to report an error message.

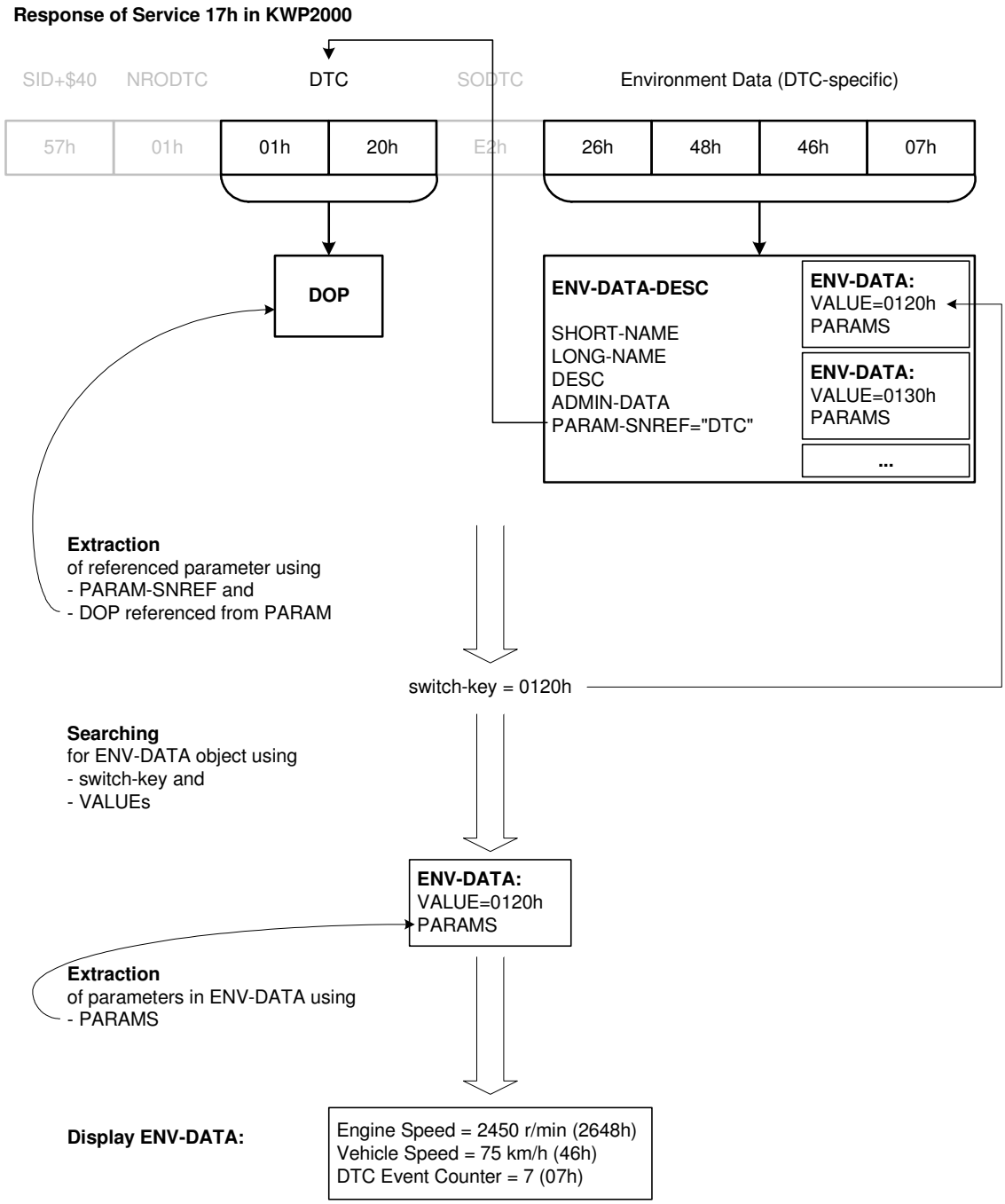


Figure 73 — Processing of ENV-DATA-DESC

EXAMPLE ENV-DATA-DESC in XML

```
<ENV-DATA-DESC ID="ENVDESC_EnvDataDesc1">
       <SHORT-NAME>ENVDESC_EnvDataDesc1</SHORT-NAME>
       <PARAM-SNREF SHORT-NAME="SwitchKeyDTC"/>
       <ENV-DATAS>
              <ENV-DATA ID="ED_1">
                     <SHORT-NAME>ED_DTC0120</SHORT-NAME>
                     <PARAMS>
                            <PARAM xsi:type="VALUE">
                                   <SHORT-NAME>EngineSpeed</SHORT-NAME>
```

```
                                <BYTE-POSITION>0</BYTE-POSITION>
                                <DOP-REF ID-REF="DOP_EngineSpeed"/>
                        </PARAM>
                        <PARAM xsi:type="VALUE">
                                <SHORT-NAME>VehicleSpeed</SHORT-NAME>
                                <BYTE-POSITION>2</BYTE-POSITION>
                                <DOP-REF ID-REF="DOP_1ByteHex"/>
                        </PARAM>
                </PARAMS>
                <DTC-VALUES>
                        <DTC-VALUE>288</DTC-VALUE>
                </DTC-VALUES>
        </ENV-DATA>
        <ENV-DATA ID="ED_2">
                <SHORT-NAME>ED_DTC0130</SHORT-NAME>
                <PARAMS>
                        <PARAM xsi:type="VALUE">
                                <SHORT-NAME>EngineSpeed</SHORT-NAME>
                                <BYTE-POSITION>0</BYTE-POSITION>
                                <DOP-REF ID-REF="DOP_EngineSpeed"/>
                        </PARAM>
                        <PARAM xsi:type="VALUE">
                                <SHORT-NAME>Voltage</SHORT-NAME>
                                <BYTE-POSITION>2</BYTE-POSITION>
                                <DOP-REF ID-REF="DOP_Voltage"/>
                        </PARAM>
                </PARAMS>
                <DTC-VALUES>
                        <DTC-VALUE>304</DTC-VALUE>
                </DTC-VALUES>
        </ENV-DATA>
        <ENV-DATA ID="ED_3">
                <SHORT-NAME>DTC2345_AND_DTC1234</SHORT-NAME>
                <PARAMS>
                        <PARAM xsi:type="VALUE">
                                <SHORT-NAME>Voltage</SHORT-NAME>
                                <BYTE-POSITION>0</BYTE-POSITION>
                                <DOP-REF ID-REF="DOP_Voltage"/>
                        </PARAM>
                        <PARAM xsi:type="VALUE">
                                <SHORT-NAME>EngineSpeed</SHORT-NAME>
                                <BYTE-POSITION>0</BYTE-POSITION>
                                <DOP-REF ID-REF="DOP_EngineSpeed"/>
                        </PARAM>
                </PARAMS>
                <DTC-VALUES>
                        <DTC-VALUE>4660</DTC-VALUE>
                        <DTC-VALUE>9029</DTC-VALUE>
                </DTC-VALUES>
        </ENV-DATA>
        <ENV-DATA-REF ID-REF="ED_4" DOCREF="LAYER_1" DOCTYPE="LAYER"/>
        <ENV-DATA ID="ED_All">
                <SHORT-NAME>ED_All</SHORT-NAME>
                <PARAMS>
                        <PARAM xsi:type="VALUE">
                                <SHORT-NAME>EventCouter</SHORT-NAME>
                                <BYTE-POSITION>3</BYTE-POSITION>
                                <DOP-REF ID-REF="DOP_1ByteHex"/>
                        </PARAM>
                </PARAMS>
                <ALL-VALUE/>
        </ENV-DATA>
    </ENV-DATAS>
</ENV-DATA-DESC>
```

In this example three DTC-specific ENV-DATA objects are defined (ID = "ED_1", "ED_*2" and "ED*_3"). Environment data with ID="ED_1" is defined for the DTC 0120h, with ID="ED_2" for 0130h, and ENV-DATA with ID="ED_3" is defined for two DTCs: 1234h and 2345h. With the line "<ENV-DATA-REF ID-REF="ED_4" DOCREF="LAYER_1" DOCTYPE="LAYER"/>" an ENV-DATA (ID = "ED_4") from another layer (with SHORT-NAME = LAYER_1) is referenced. This ENV-DATA is used at run time as if it was defined in place. Additionally, an ENV-DATA is defined that is valid for all DTCs (ID = ED_All). In this example, Byte 3 is always an event counter, which gives how often the error has occurred.

Rules for authoring

Environmental data common for all DTCs should be specified inside a single ENV-DATA with the flag ALL-VALUE. All environmental data specific for one or more DTCs must be specified inside a single ENV-DATA with an explicit specification of the corresponding DTCs. This means that at most one ENV-DATA with the flag ALL-VALUE may exist.
An ENV-DATA-DESC may only be referenced by a response parameter of a service whose DIAGNOSTIC-CLASS attribute indicates an error service (DIAGNOSTIC-CLASS="FAULTREAD"). Furthermore, the response must contain exactly one parameter whose short-name matches the SHORT-NAME attribute of PARAM-SNREF in the ENV-DATA-DESC object.

Reuse of ENV-DATA objects

In analogy to DTC objects, ENV-DATA objects can also be reused. If new ENV-DATAs are to be added to the ENV-DATA list of the ENV-DATA-DESC object (in the figure below the ENV-DATA-DESC C), a new ENV-DATA-DESC object D is to be created, which references the ENV-DATAs, taken over from the object C and defines the new ENV-DATAs that are to be added. Not used ENV-DATAs are not referenced. The figure below shows both use cases. The ENV-DATA structure for the diagnostic trouble code 0120h is reused in the ECU-VARIANT layer by referencing to it from ENV-DATA-DESC D. The ENV-DATA for the DTC 0130h is not referenced and is thus not available at the lower layer. Additionally, defining a new ENV-DATA object for the DTC 0131h shows a possibility for extension of ENV-DATA-DESC.

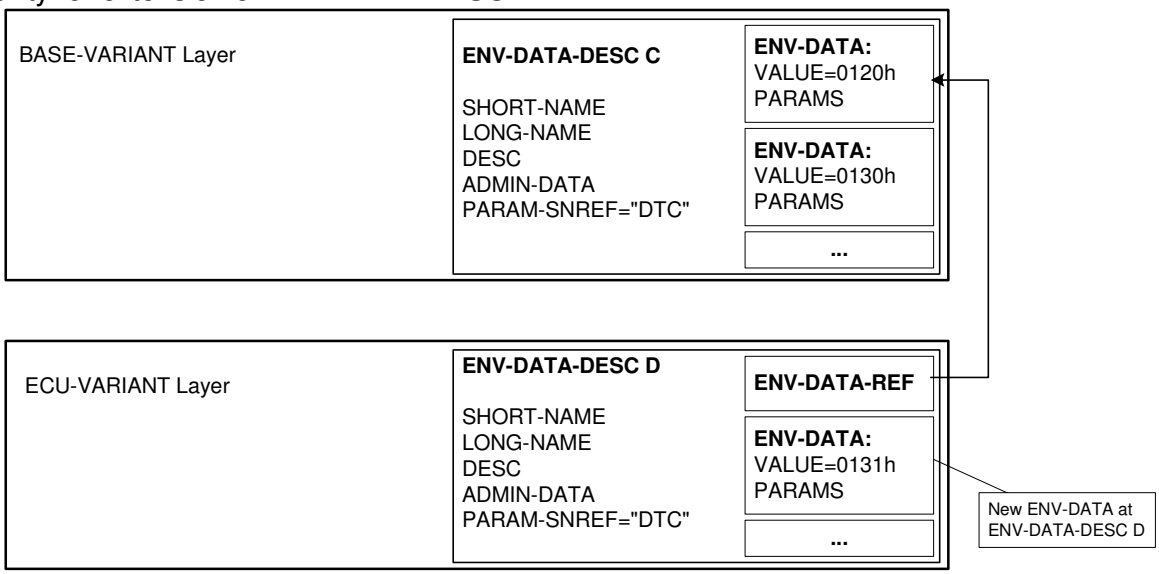


Figure 74 — Reuse of ENV-DATA

7.3.6.10 COMPLEX DATA (COMPLEX DATA OBJECT PROPERTY)

7.3.6.10.1 Overview

Complex diagnostic data object properties (DOPs) are used to define and interpret complex ECU responses (or structured parts of a tester request) containing multiple data items, which are grouped in a structure, a list or a field. This is also needed to describe ECU responses which are dynamic i.e. the size and actual structures that are known only at runtime. Additionally, there is a complex DOP, the so called multiplexer (MUX), which is used to interpret data depending on a switch key. Most complex DOPs own an ID attribute used for referencing and the attribute IS-VISIBLE. The latter can be set to "true" or "false". Set to "true", this attribute causes the MCD 3D / MVCI Diagnostic Server API to show the structural information given by this complex DOP and it is not shown otherwise. Like other DOPs, complex DOPs have the standard elements SHORT-NAME, LONG-NAME, DESC and ADMIN-DATA. In the following all types of complex DOPs are described in detail. Complex DOPs are referenced from a PARAM of a response in the same way like simple DOPs. The BYTE- and BIT-POSITIONs of a PARAM are given there as absolute positions in the PDU if the PARAM is directly part of a POS-RESPONSE, NEG-REPONSE, GLOBAL-NEG-RESPONSE or REQUEST. In contrast to that, the BYTE-POSITION of a PARAM used by a STRUCTURE is always given relatively to the start of this STRUCTURE (see figure below). In a description of an ECU response a PARAM of type VALUE at absolute byteposition 2 is defined that refers to a complex DOP of type STRUCTURE (description will follow below). Inside the STRUCTURE two PARAM at relative the bytepositions 1 and 3 are defined.

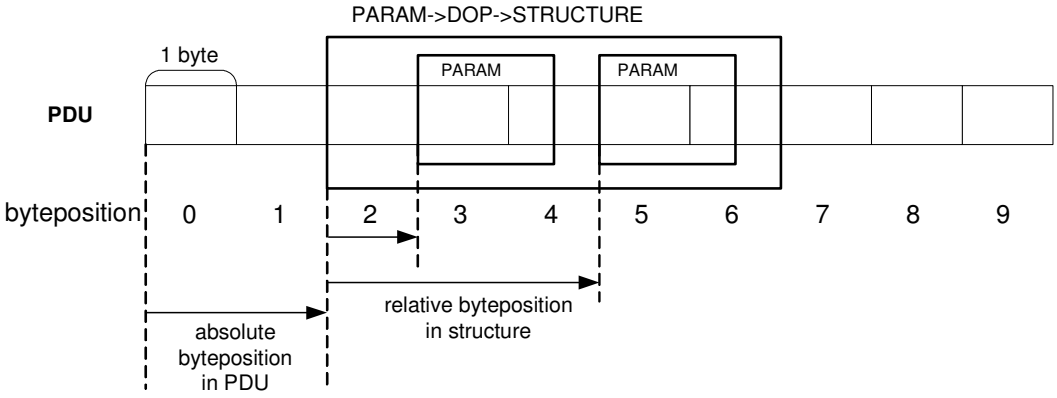


Figure 75 — Absolute and relative byteposition

Drawing: ComplexData
Package: DataParameter::ComplexData

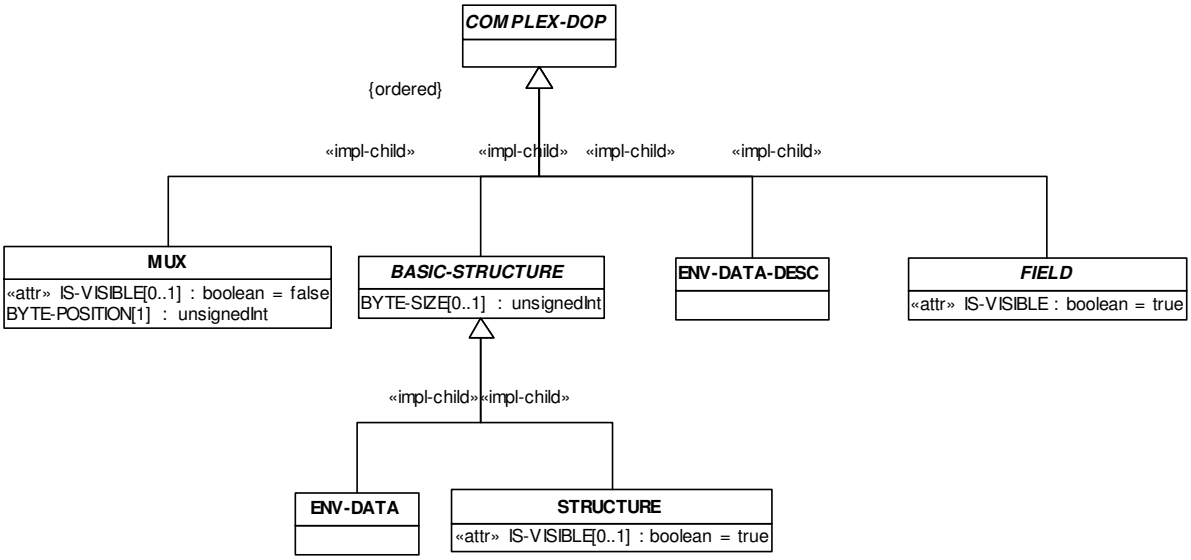


Figure 76 — Complex data

7.3.6.10.2 STRUCTURE

A STRUCTURE is some kind of wrapper to enable recursion that combines several PARAMs into a group of parameters belonging together. This is possible for REQUEST and RESPONSE. Beside information structuring, the use of STRUCTUREs enables an easy reuse of parameter groups. The BIT-POSITION of a PARAM using a STRUCTURE directly or via any kind of COMPLEX-DOP must be equal to "0" i.e. a STRUCTURE starts always at byte edge. For example, the status of DTC might be defined as a STRUCTURE containing a DTC Warning Lamp Calibration Status, a DTC Storage Status, a DTC Readiness Flag and a DTC Fault Symptom. In ISO 14230 this structure can then be used both at service $17 and at service $18.

Drawing: Structure
Package: DataParameter::ComplexData

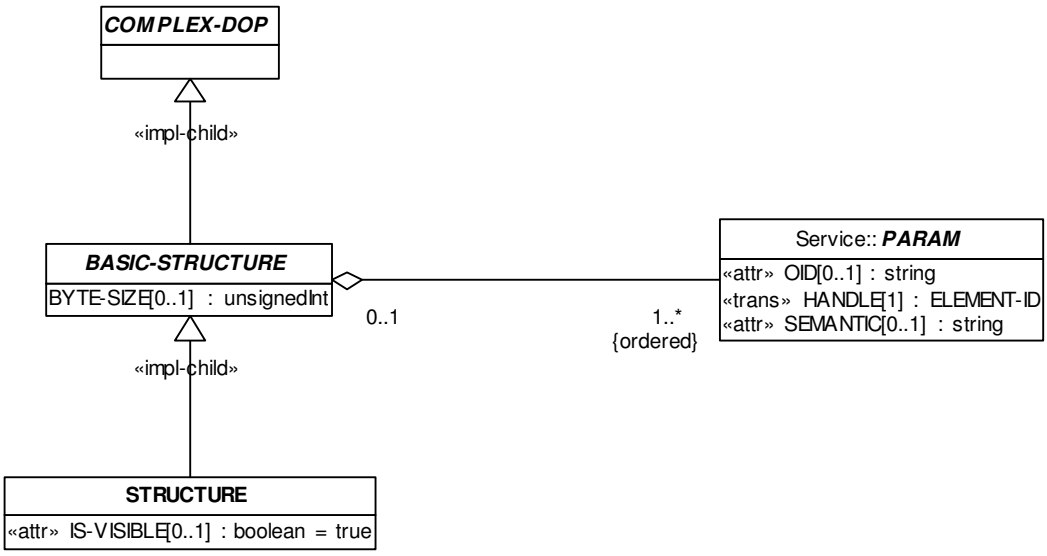


Figure 77 — Structure

Within a STRUCTURE one or more PARAMs are defined in the same way like within a response. Each PARAM represents an item which is a member of the group built by this STRUCTURE. The optional attribute BYTE-SIZE gives the size of the whole structure in bytes. It must not be less than the total size of the included parameters. It can be greater than the total size of the included items if the developer aims to create space after the last item. If the STRUCTURE has any dynamic components (e.g. MUX or any FIELD with dynamic length) the BYTE-SIZE cannot and must not be given.
Since a complex DOP can contain PARAMs, which themselves can use complex DOPs, it is possible to define complex data structures recursively. The flag IS-VISBLE is used to define if the STRUCTURE that is used as a wrapper should be visible at the MCD 3D / MVCI server. If e.g. a STRUCTURE S2 consists of parameter PARAM_2A, a parameter PARAM_2B and the attribute IS-VISIBLE="true" this results in a structure of name S2 at the application interface (that contains "2A" and "2B"). If the attribute IS-VISIBLE="false" is set, no structure but simply two result parameters "2A" and "2B" are returned.

EXAMPLE Recursive definition of data structures using STRUCTURE

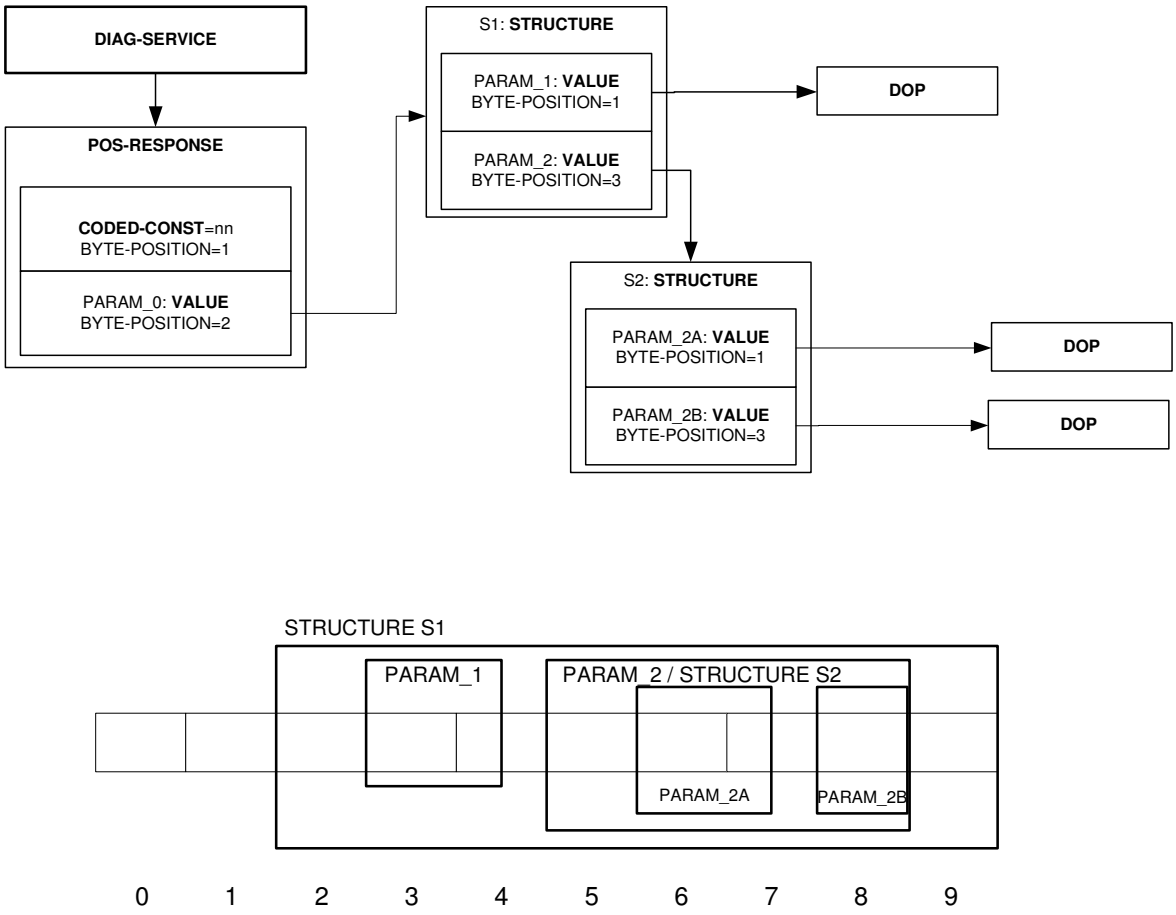


Figure 78 — Recursive definition of data structures using STRUCTURE

The structure S1 looks in XML as follows:

```
<STRUCTURE ID="STRUCT_S1" IS-VISIBLE="false">
	<SHORT-NAME>S1</SHORT-NAME>
	<BYTE-SIZE>8</BYTE-SIZE>
	<PARAMS>
		<PARAM xsi:type="VALUE" SEMANTIC="DATA">
			<SHORT-NAME>PARAM_1</SHORT-NAME>
			<BYTE-POSITION>1</BYTE-POSITION>
			<DOP-REF ID-REF="DOP-ID"/>
		</PARAM>
		<PARAM xsi:type="VALUE" SEMANTIC="DATA">
			<SHORT-NAME>PARAM_2</SHORT-NAME>
			<BYTE-POSITION>3</BYTE-POSITION>
			<DOP-REF ID-REF="STRUCT_S2"/>
		</PARAM>
	</PARAMS>
</STRUCTURE>
```

EXAMPLE The following structure implements the example mentioned above. It represents the status of DTC in ISO 14230 containing a DTC Warning Lamp Calibration Status, a DTC Storage Status, a DTC Readiness Flag and DTC Fault Symptom in one byte.

```
<STRUCTURE ID="STRUCT_SODTC" IS-VISIBLE="false">
	<SHORT-NAME>SODTC</SHORT-NAME>
	<LONG-NAME>Status of DTC</LONG-NAME>
	<BYTE-SIZE>1</BYTE-SIZE>
```

```
      <PARAMS>
            <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                  <SHORT-NAME>DTC_WLCS</SHORT-NAME>
                  <LONG-NAME>DTC Warning Lamp Calibration Status</LONG-NAME>
                  <BYTE-POSITION>0</BYTE-POSITION>
                  <BIT-POSITION>7</BIT-POSITION>
                  <DOP-REF ID-REF="DOP_DTC_WLCS"/>
            </PARAM>
            <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                  <SHORT-NAME>DTC_SS</SHORT-NAME>
                  <LONG-NAME>DTC Storage Status</LONG-NAME>
                  <BYTE-POSITION>0</BYTE-POSITION>
                  <BIT-POSITION>5</BIT-POSITION>
                  <DOP-REF ID-REF="DOP_DTC_SS"/>
            </PARAM>
            <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                  <SHORT-NAME>DTC_RF</SHORT-NAME>
                  <LONG-NAME>DTC Readyness Flag</LONG-NAME>
                  <BYTE-POSITION>0</BYTE-POSITION>
                  <BIT-POSITION>4</BIT-POSITION>
                  <DOP-REF ID-REF="DOP_DTC_RF"/>
            </PARAM>
            <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                  <SHORT-NAME>DTC_FS</SHORT-NAME>
                  <LONG-NAME>DTC Fault Symptom</LONG-NAME>
                  <BYTE-POSITION>0</BYTE-POSITION>
                  <BIT-POSITION>0</BIT-POSITION>
                  <DOP-REF ID-REF="DOP_DTC_FS"/>
            </PARAM>
      </PARAMS>
</STRUCTURE>
```

7.3.6.10.3 STATIC-FIELD

STATIC-FIELDs are used when the PDU contains a recurring structure and the number of repetitions is fixed and does not have to be determined dynamically. In other words, a STATIC-FIELD is used to describe a repetition of a STRUCTURE with an a priori fixed number of repetitions.

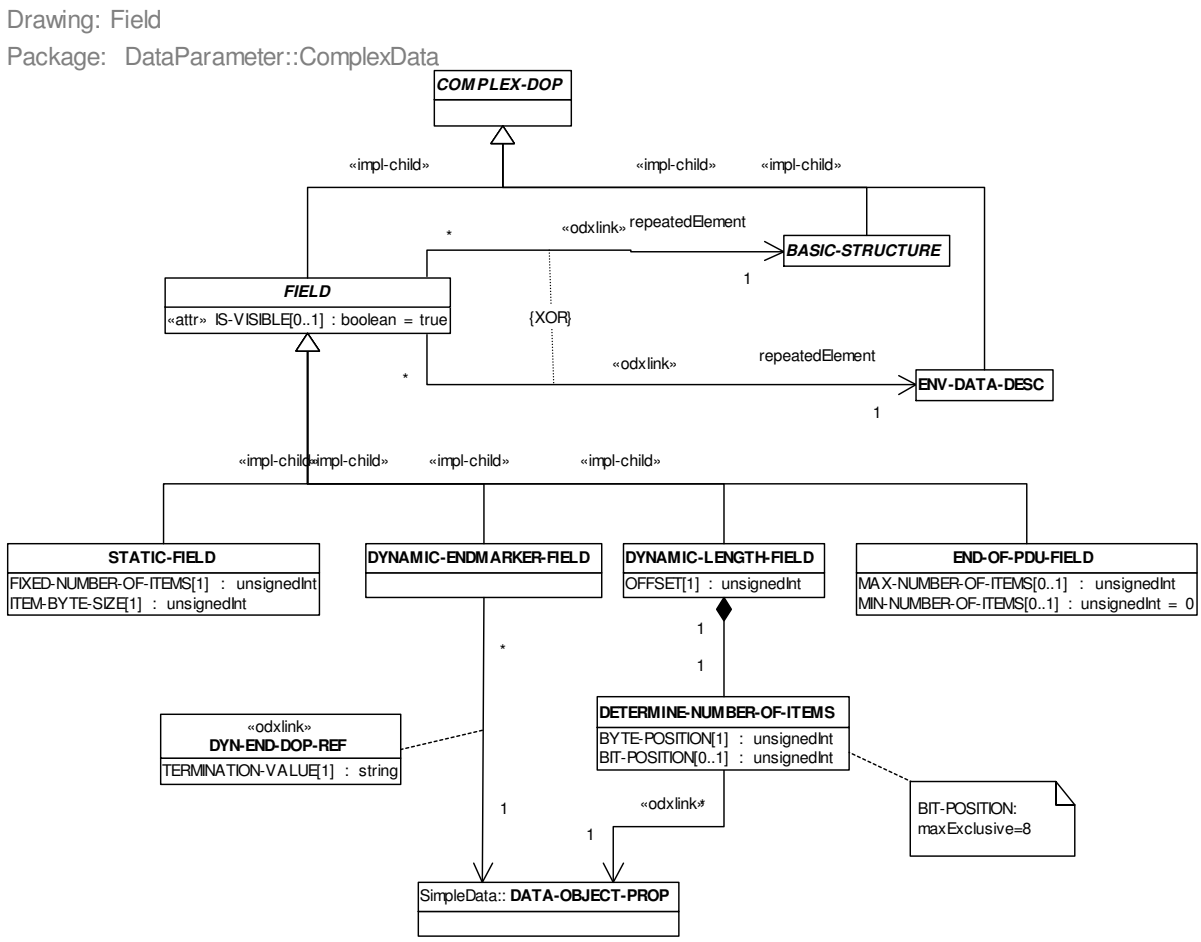


Figure 79 — Field

The number of repetitions is given via the attribute FIXED-NUMBER-OF-ITEMS. ITEM-BYTE-SIZE determines the length of one item in the field. It is given in bytes and must not be less than the length of one item. A reference to a BASIC-STRUCTURE defines the structure to be repeated (e.g. a bitfield of binary flags). The flag IS-VISIBLE is used to define if the list itself should be visible at the MCD 3D / MVCI Diagnostic Server API.

NOTE It must be ensured that the items have a fixed bytelength i.e. must not be of dynamic type.

EXAMPLE In the following example a STATIC-FIELD is defined, which contains two elements with the structure defined by the object with ID=" STRUCT_S_1":

```
<STATIC-FIELD ID="SF-1">
      <SHORT-NAME>SF-1</SHORT-NAME>
      <BASIC-STRUCTURE-REF ID-REF="STRUCT_S-1"/>
      <FIXED-NUMBER-OF-ITEMS>2</FIXED-NUMBER-OF-ITEMS>
      <ITEM-BYTE-SIZE>4</ITEM-BYTE-SIZE>
</STATIC-FIELD>
```

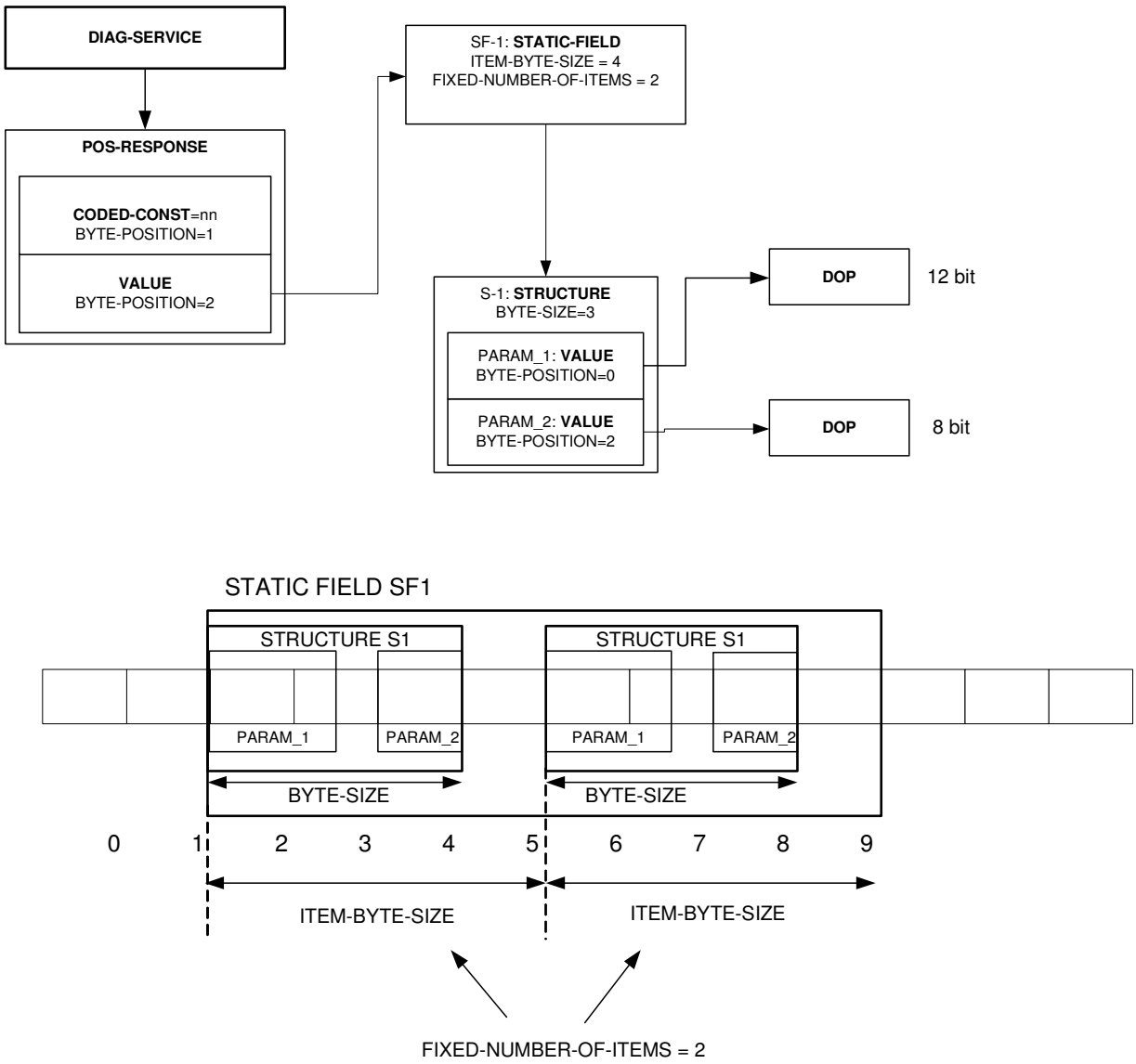


Figure 80 — Using STATIC-FIELD

7.3.6.10.4 DYNAMIC-LENGTH-FIELD

DYNAMIC-LENGTH-FIELDs are fields of items (STRUCTUREs) with variable number of repetitions, which can be determined only at runtime.
The number of repetitions is given via DETERMINE-NUMBER-OF-ITEMS. BYTE-POSITION, BIT-POSITION and DOP-REF are used there to calculate the integer value. The DOP referenced here must result in an unsigned integer datatype. The STRUCTURE-REF is used to specify the structure to be repeated.

EXAMPLE In the following example a DYNAMIC-LENGTH-FIELD is defined, which contains a field of elements with the structure defined by the object with ID="STRUCT_S-1". The DOP with ID="DOP_1" extracts a number from the PDU at BYTE-POSITION=0 and calculates the element count.

```
<DYNAMIC-LENGTH-FIELD ID="DLF1" IS-VISIBLE="true">
       <SHORT-NAME>DLF1</SHORT-NAME>
       <BASIC-STRUCTURE-REF ID-REF="STRUCT_S-1"></BASIC-STRUCTURE-REF>
       <OFFSET>2</OFFSET>
       <DETERMINE-NUMBER-OF-ITEMS>
              <BYTE-POSITION>0</BYTE-POSITION>
              <DATA-OBJECT-PROP-REF ID-REF="DOP_1"/>
       </DETERMINE-NUMBER-OF-ITEMS>
</DYNAMIC-LENGTH-FIELD>
```

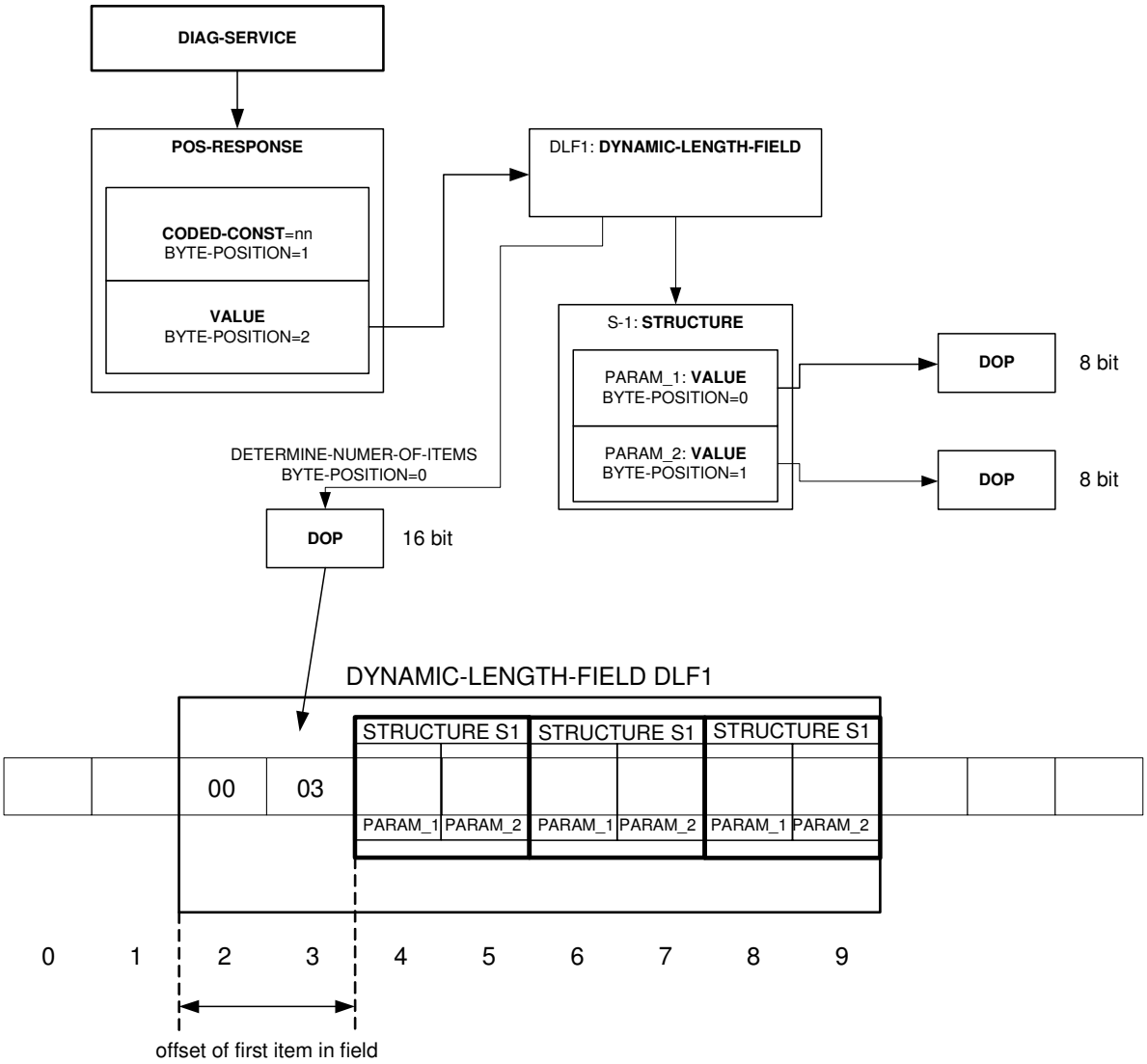


Figure 81 — Using DYNAMIC-LENGTH-FIELD

7.3.6.10.5 DYNAMIC-ENDMARKER-FIELD

DYNAMIC-ENDMARKER-FIELDs are fields of a STRUCTURE, which is repeated until an end-marker is found (the TERMINATION-VALUE). Before each iteration step the referenced DOP is used to calculate a physical value of the parameter at the current position in the PDU. If the resulting physical value matches the TERMINATION-VALUE, the field ends without any additional item.
DOP-REF is used to reference the DOP, which calculates the physical value of the parameter at the current position in the PDU. TERMINATION-VALUE is the end-marker given as a physical value. STRUCTURE-REF is used to specify the structure to be repeated.
In the following example a DYNAMIC-ENDMARKER-FIELD is defined, which contains a field of elements with the structure defined by the object with ID="STRUCT_S1". Before each iteration the DOP with ID="DOP_1" extracts a number from the PDU at the current position which is compared with TERMINATION-VALUE=255. The iteration is stopped when the value 255 is reached.
EXAMPLE

```
<DYNAMIC-ENDMARKER-FIELD ID="DEF1" IS-VISIBLE="true">
       <SHORT-NAME>DEF1</SHORT-NAME>
       <BASIC-STRUCTURE-REF ID-REF="STRUCT_S1"/>
       <DATA-OBJECT-PROP-REF ID-REF="DOP_1">
              <TERMINATION-VALUE>255</TERMINATION-VALUE>
       </DATA-OBJECT-PROP-REF>
</DYNAMIC-ENDMARKER-FIELD>
```

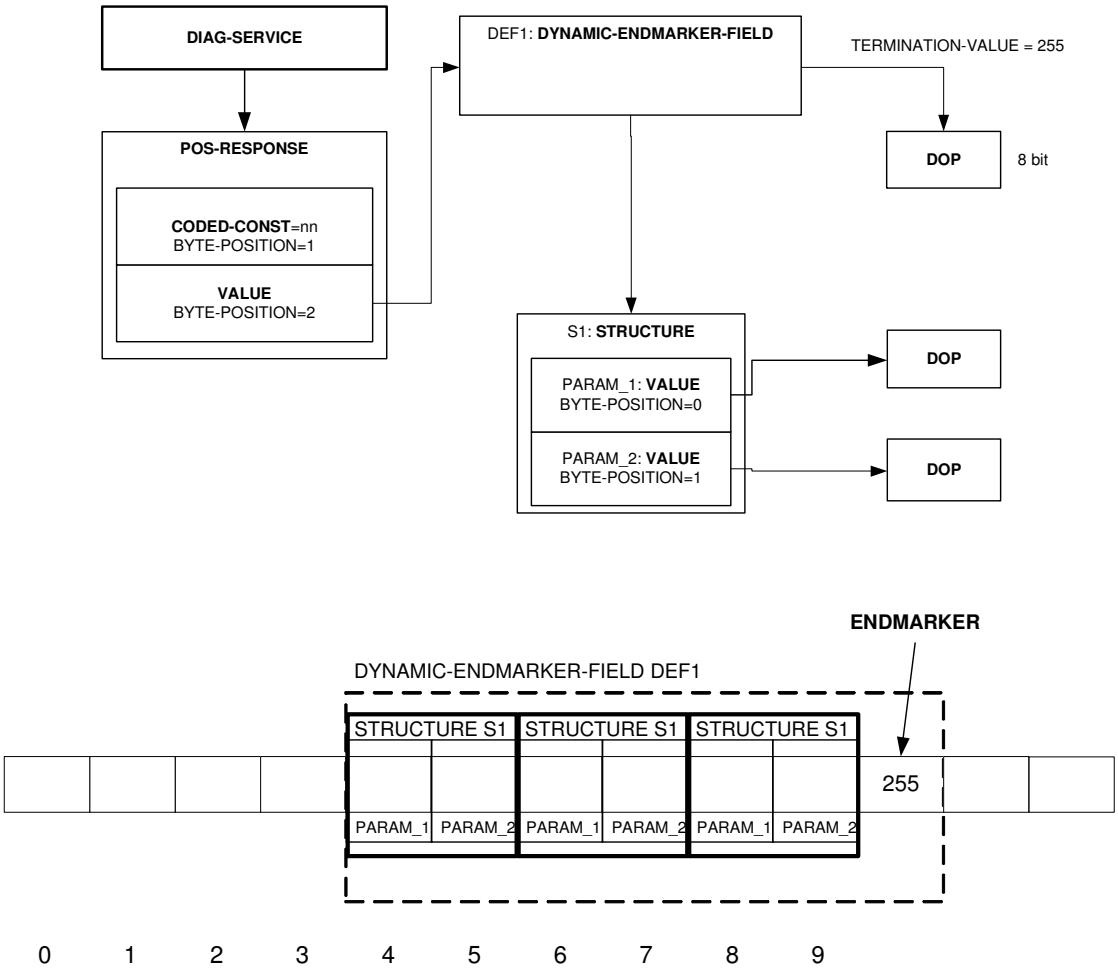


Figure 82 — Using DYNAMIC-ENDMARKER-FIELD

7.3.6.10.6 END-OF-PDU-FIELD

END-OF-PDU-FIELDs are similar to DYNAMIC-ENDMARKER-FIELDs with the difference, that one item is repeated until the end of the PDU (or as often as set by the application in the case of using this field in a description of a request). The definition of MAX-NUMBER-OF-ITEMS and MIN-NUMBER-OF-ITEMS makes it possible to define ranges of the dynamic case when this field is used in a testers request message. In the case of describing an ECUs response, the end of the PDU is some kind of "endmarker" as described above. The values of MAX-NUMBER-OF-ITEMS and MIN-NUMBER-OF-ITEMS are ignored in that case. In the case of describing a tester's request, the application must set the number of items (within the bounds defined by MIN/MAX-NUMBER-OF-ITEMS).
In addition to common attributes and elements of complex DOPs, END-OF-PDU-FIELDs simply have a reference to a STRUCTURE, which describes the structure to be repeated.
EXAMPLE

```
<END-OF-PDU-FIELD ID="EOP1" IS-VISIBLE="false">
      <SHORT-NAME>EOP1</SHORT-NAME>
      <BASIC-STRUCTURE-REF ID-REF="STRUCT_S1"></BASIC-STRUCTURE-REF>
      <MAX-NUMBER-OF-ITEMS>6</MAX-NUMBER-OF-ITEMS>
      <MIN-NUMBER-OF-ITEMS>1</MIN-NUMBER-OF-ITEMS>
</END-OF-PDU-FIELD>
```

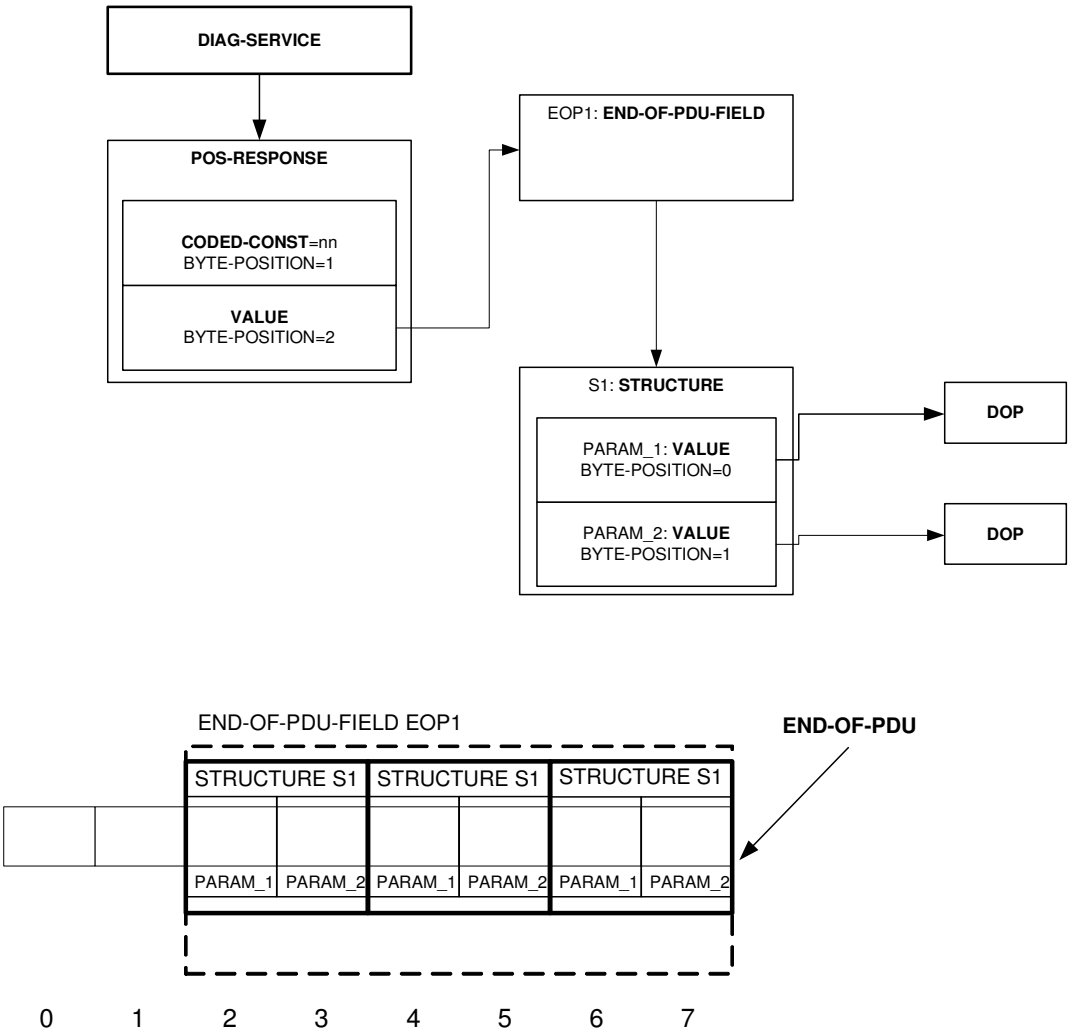


Figure 83 — Using END-OF-PDU-FIELD

7.3.6.10.7 MUX

Multiplexers (MUX) are used to interpret data stream depending on the value of a switch-key (similar to switch-case statements in programming languages like C or Java).

Drawing: Mux
Package: DataParameter::ComplexData

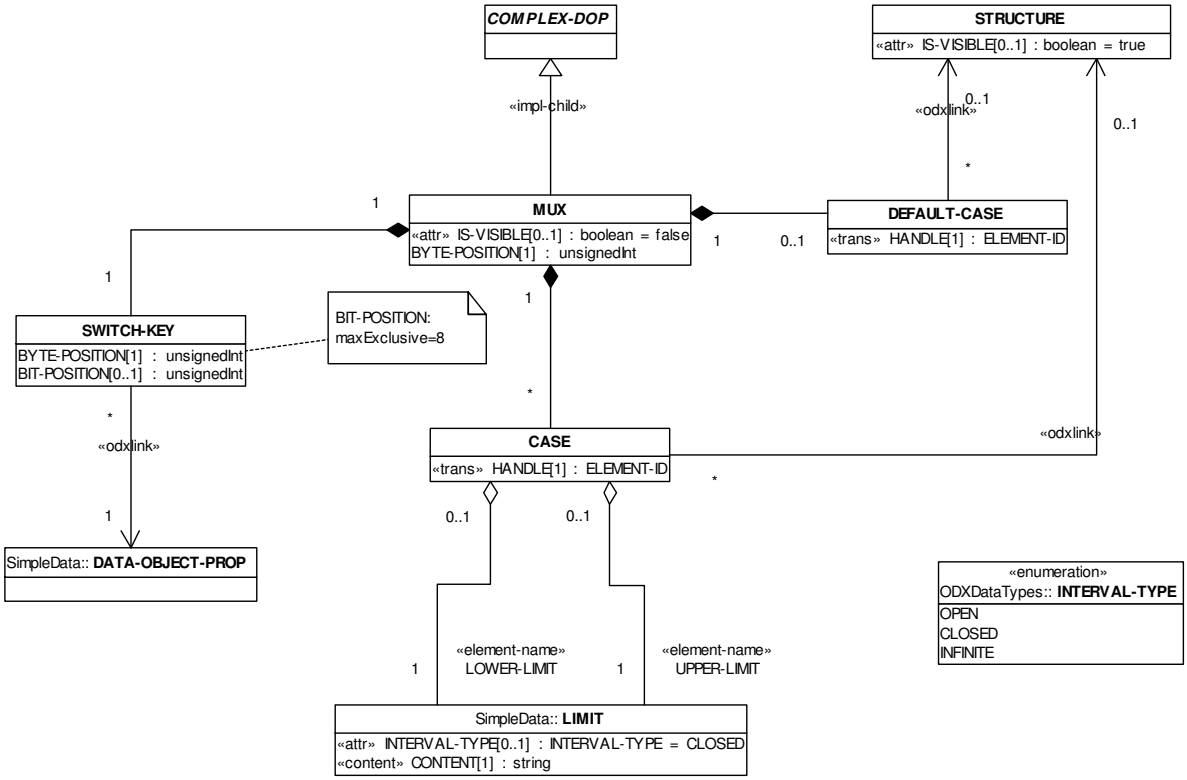


Figure 84 — Multiplexer / MUX

A PARAM using a MUX must not have a BIT-POSITION because a MUX always starts at byte edge. Each MUX contains the definition of a BYTE-POSITION, SWITCH-KEY and a set of CASE definitions. The BYTE-POSITION defines the byte offset of the data (described within a CASE) relatively to the start position of the MUX (position of the current parameter).
BIT- and BYTE-POSITION at SWITCH-KEY define the position of the switch-key in the PDU relatively to the start position of the MUX. Each SWITCH-KEY references a DATA-OBJECT-PROP, which calculates the extracted value into its physical representation. The physical value is then used for matching a CASE interval.
A CASE consists of a SHORT-NAME, a LONG-NAME, a reference to a STRUCTURE and a range defined by LIMIT (lower and upper, compare COMPU-SCALE). The case is valid if the SWITCH-KEY is within the range defined by LOWER-LIMIT/UPPER-LIMIT. There must be no overlap of intervals. Additionally, a DEFAULT-CASE might be declared which is used when no case matches. In contrast to a CASE, the DEFAULT-CASE contains only a SHORT-NAME, a LONG-NAME and references a STRUCTURE. Every STRUCTURE referenced by CASE or DEFAULT-CASE must be defined with IS-VISIBLE="false". The DEFAULT-CASE may be omitted if the runtime system shall be forced to report an error when no case matches.

EXAMPLE The following example code defines a MUX, which interprets the data depending on the byte before the parameter which references this MUX.

```
<MUX ID=" M1">
       <SHORT-NAME> M1</SHORT-NAME>
       <BYTE-POSITION>1</BYTE-POSITION>
       <SWITCH-KEY>
              <BYTE-POSITION>0</BYTE-POSITION>
              <DATA-OBJECT-PROP-REF ID-REF="DOP_1"/>
       </SWITCH-KEY>
       <CASES>
              <CASE>
                     <SHORT-NAME>case_1</SHORT-NAME>
                     <STRUCTURE-REF ID-REF=" SCASE-24"/>
                     <LOWER-LIMIT>24</LOWER-LIMIT>
                     <UPPER-LIMIT>24</UPPER-LIMIT>
              </CASE>
              <CASE>
                     <SHORT-NAME>case_2</SHORT-NAME>
                     <STRUCTURE-REF ID-REF=" SCASE-25"/>
                     <LOWER-LIMIT>25</LOWER-LIMIT>
                     <UPPER-LIMIT>25</UPPER-LIMIT>
              </CASE>
       </CASES>
</MUX>
```

The figure below clarifies the circumstances. The third byte of the shown PDU is the switch-key. In the first step the switch-key is extracted using BYTE-POSITION and the DOP referenced by SWITCH-KEY. In the second step the matching CASE is searched. This appropriate structure is then used to interpret the rest of the message. The physical type of the DOP used by the SWITCH-KEY must match the data type in LOWER-LIMIT and UPPER-LIMIT.

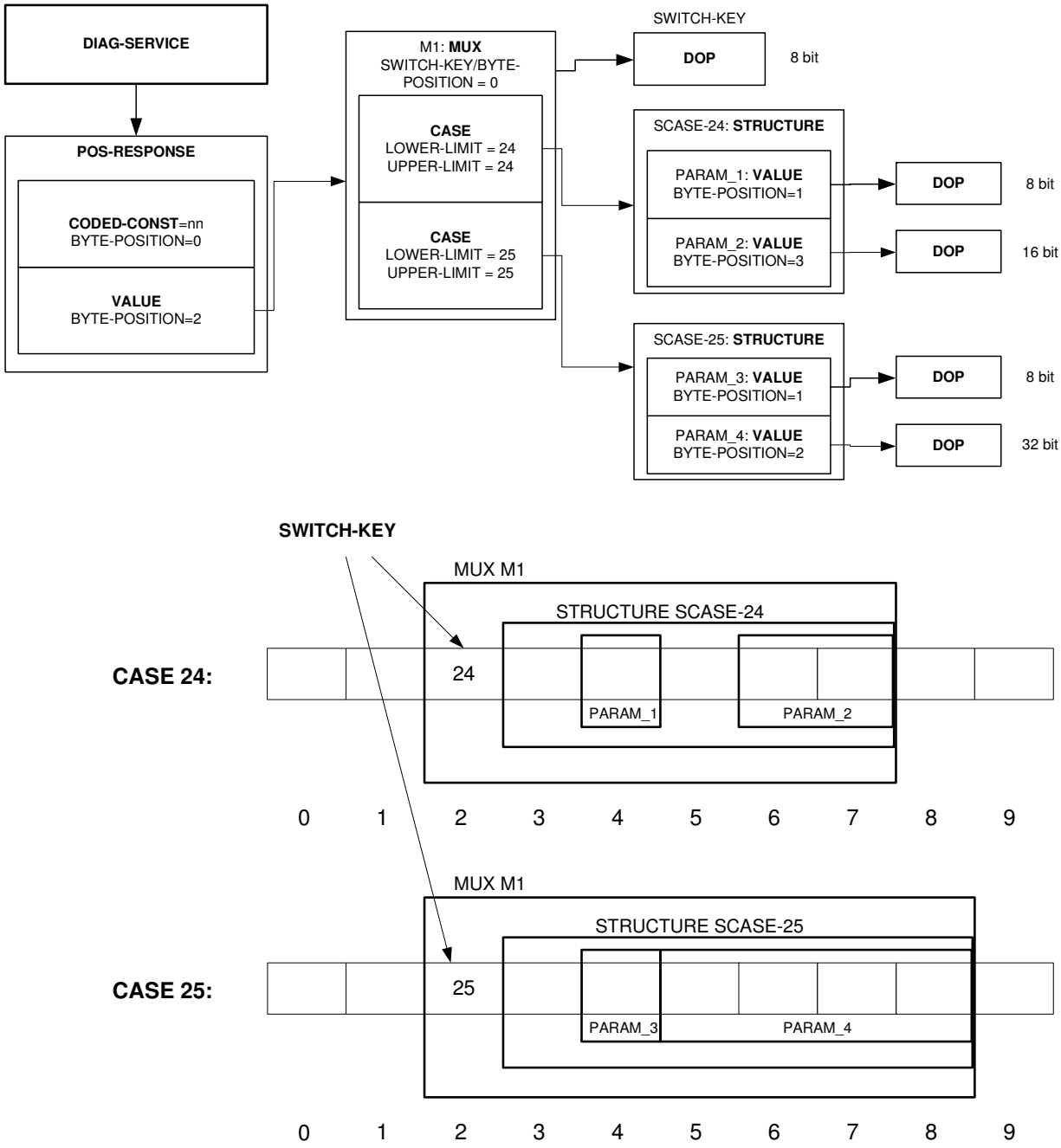


Figure 85 — Using MUX

7.3.6.11 TABLE – COMPOUND DATA OBJECT

The concept of data identifiers or parameter identifiers describes the association of a data structure definition to a numerical identifier. This data structure may contain a single parameter or a list of parameters. Typically there is a single parameter in case of analog values like sensor voltages or a list of parameters in case of discrete values like switch statuses.
Most protocols use data identifier to describe data structures contained in response messages, however ISO14229-1 defines for example the InputOutputControlByIdentifier (2Fh) service or WriteDataByIdentifier (2Eh), which extends the use of DataIdentifier also

to specify data structures contained in request messages. For this proceeding the element TABLE can be used.

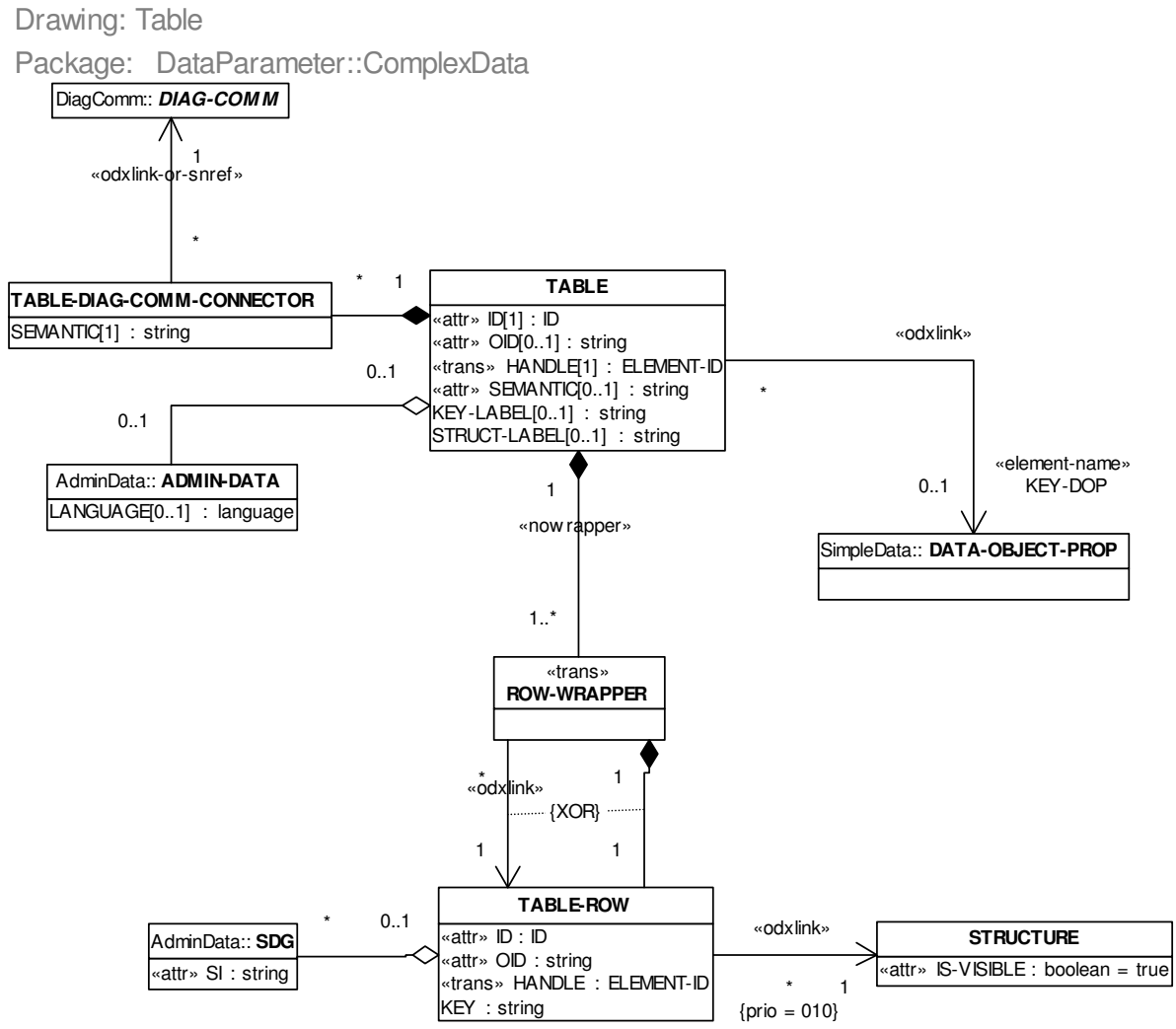


Figure 86 — TABLE

The usage of TABLE is similar to the DOP structure and is placed as a substructure of DIAG-DATA-DICTIONARY-SPEC.
Anyhow the TABLE is not derived from DOP-BASE because it can be used at the param type TABLE-KEY only unlike DOP-BASE-derived objects. The TABLE can be used for different kind of KEYs (data identifier). On account of this it has a SEMANTIC attribute. Values of SEMANTIC may be DATA-ID (DataIdentifier), PACKAGE-ID (PackageIdentifier), PARAMETER-ID (ParameterIdentifier), LOCAL-ID (LocalIdentifier), COMMON-ID (CommonIdentifier) and ADDRESS (MemoryAddress in the ECU) for example.
According to its name this structure is a simple table composed of a KEY-column and a STRUCT-column. The columns are labelled by KEY-LABEL and STRUCT-LABEL for the documentation use-case. A TABLEROW associates a data identifier called KEY with a parameter definition via reference on a STRUCTURE. A TABLE-ROW already defined in another TABLE can be reused by reference.

The DATA-OBJECT-PROP referenced by the TABLE defines the coding of the KEY in any request parameter and the extraction of the KEY out of any response parameter. The KEY value in the XML-instance is always a "physical" value.
Three PARAM types are introduced for usage with TABLE:
TABLE-KEY: A parameter of type TABLE-KEY is the part of the PDU where the KEY (data identifier) has to be read in the case of response or written in the case of request. The parameter can refer the TABLE for dynamic parameter definition or it refers a TABLE-ROW for static definition similar to parameter type PHYS-CONST. The TABLE-KEY-parameter in the response shows the position where the KEY should be extracted using the DATA-OBJECT-PROP referenced by TABLE. The physical result must match with exactly one KEY-value in the TABLE. The TABLE-ROW containing the matching KEY refers the STRUCTURE necessary to the data extraction of all TABLESTRUCT-parameters that belongs to the TABLE-KEY-parameter.
TABLE-STRUCT: This parameter type defines the position of the data to be extracted by using the TABLEROW-referenced STRUCTURE. The corresponding TABLE-KEY-parameter is referenced by an odx-link.
TABLE-ENTRY: This parameter type should be used for grouping of TABLE-ROWsthat is multiple-stage structuring of data. TABLE-ENTRY has an attribute TARGET to indicate whether the ID is also part of the common data structure. The compound data structure is a sequence without a gap. This parameter type can only be used in a structure referenced by a TABLE.

awing: Parameter
ckage: Datastream::Service

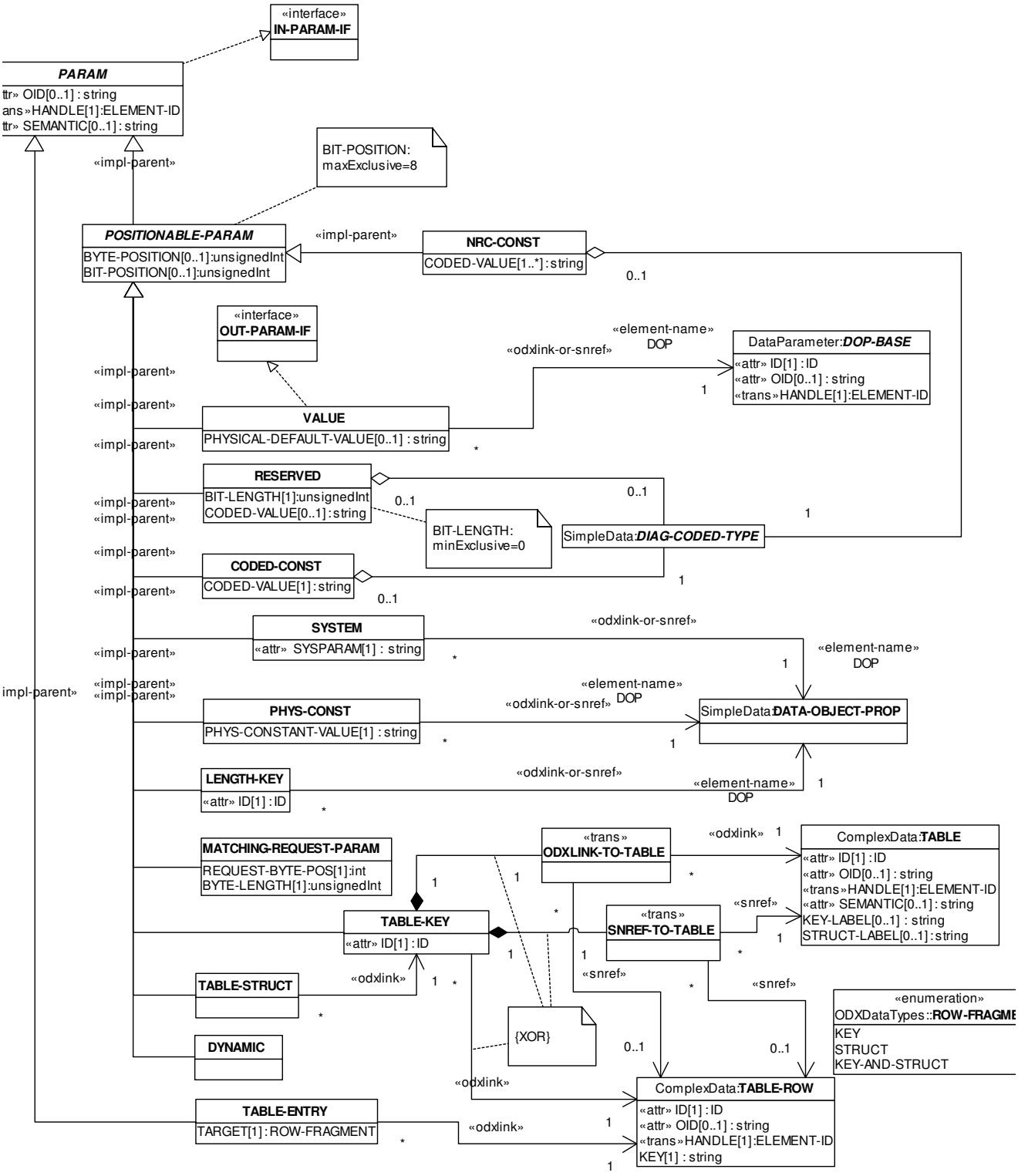


Figure 87 — Parameter

EXAMPLE In the following figures some examples for writing and reading data identifier are shown. Two of them are implemented in static and dynamic manner. In case of dynamic implementation of a request parameter a TABLE-KEY refers a TABLE as a whole. Identical to PARAM-type VALUE the tester can choose a TABLE-ROW during runtime.

At the static implementation the TABLE-KEY refers a TABLE-ROW directly and defined a permanent parameter value in this way identical to PARAM-type PHYS-CONST.

Drawing: OBD - Service $01

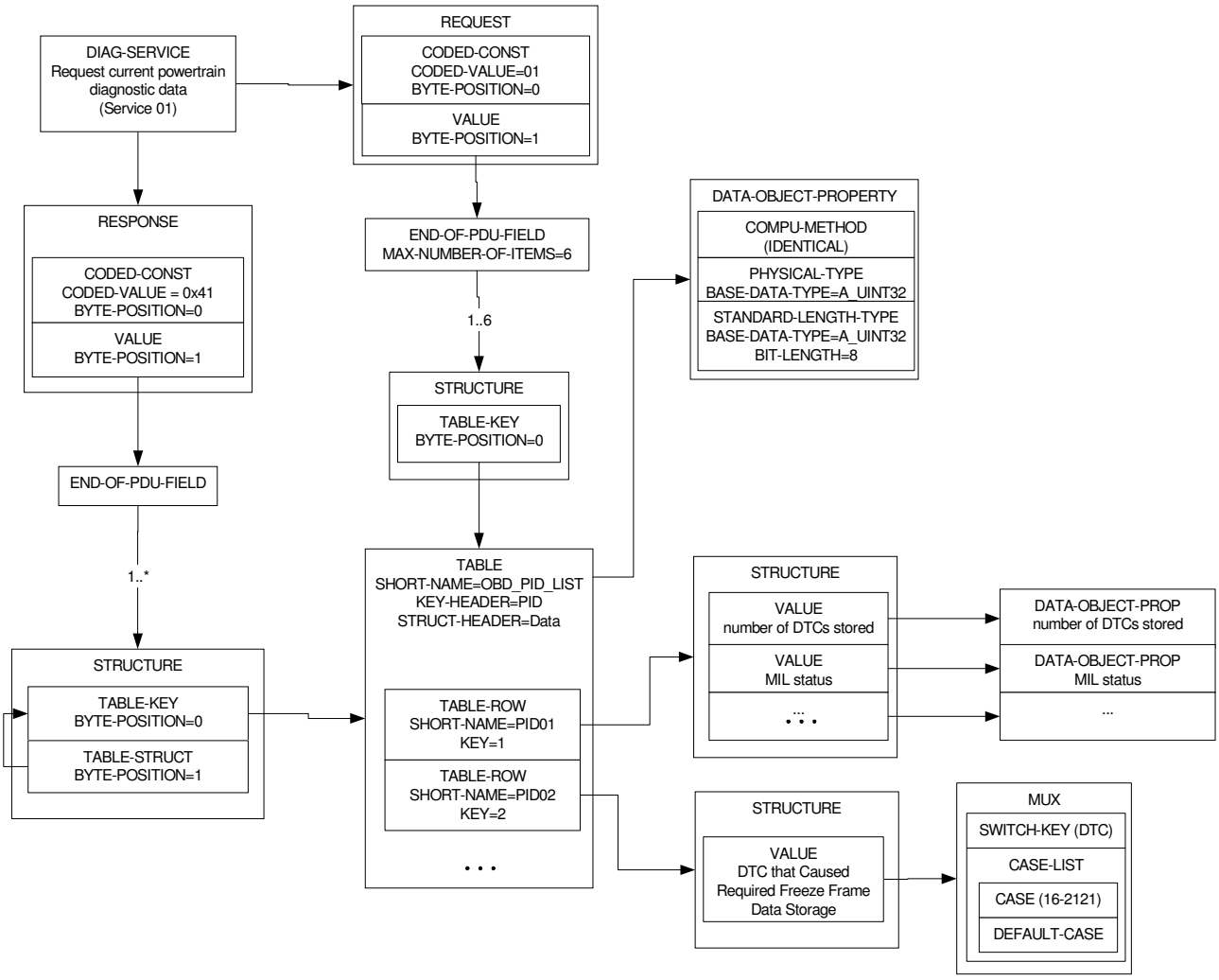


Figure 88 — OBD PID example

Drawing: ISO14229-1 RDBI (dyn)

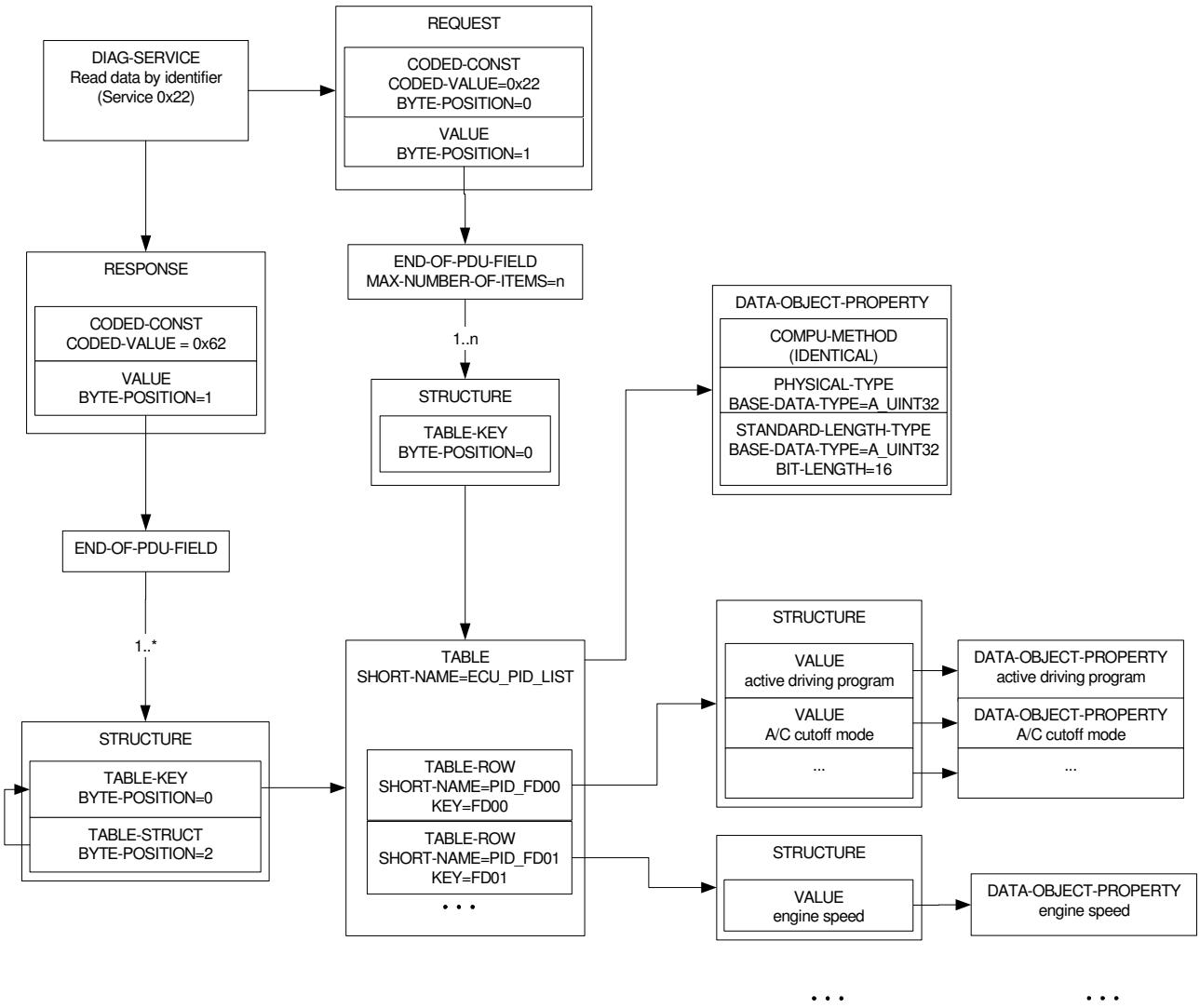


Figure 89 — ISO 14229-1 Read data by identifier example (dynamic)

Drawing: ISO14229-1 RDBI (static)

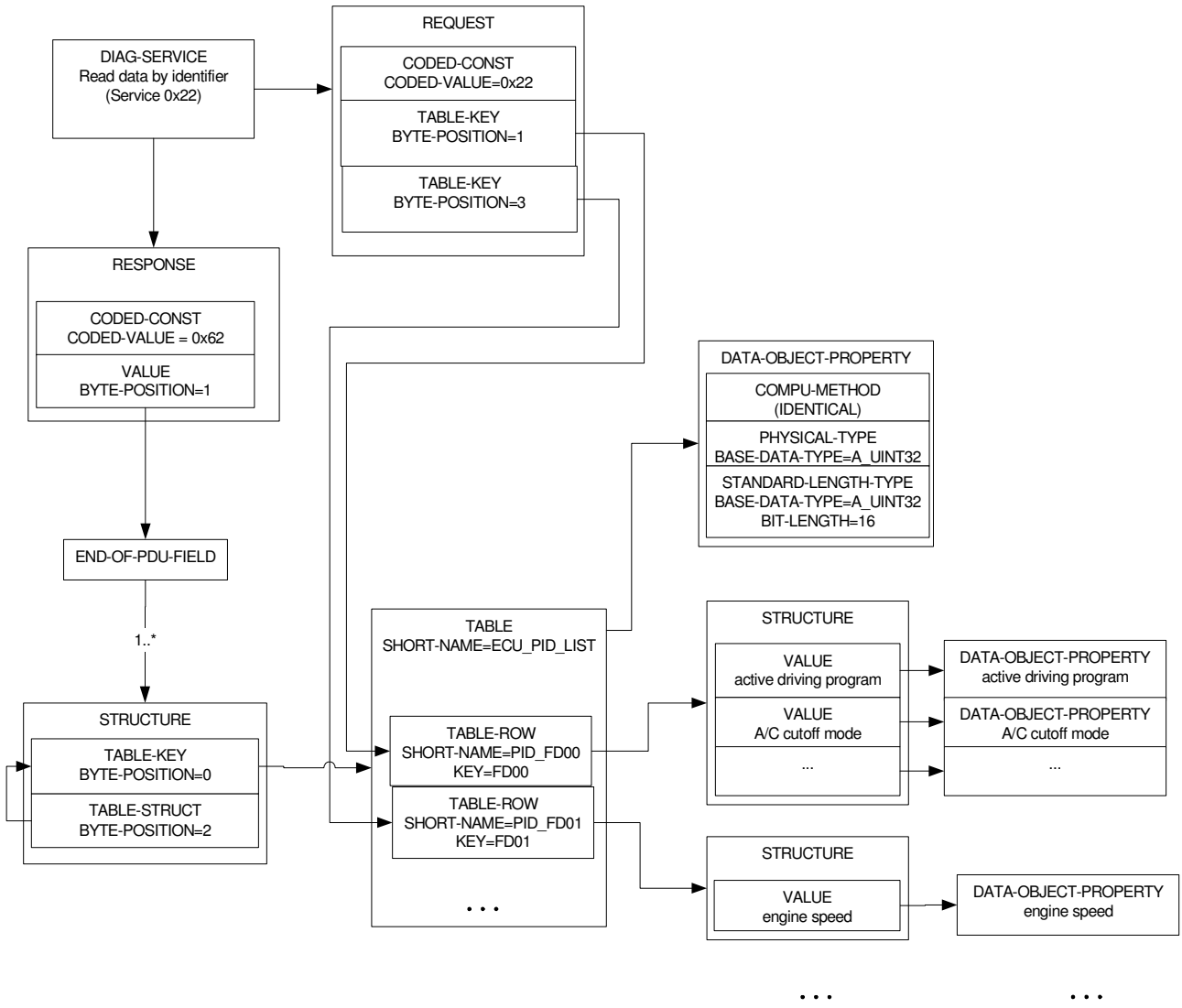


Figure 90 — ISO 14229-1 Read data by identifier example (static)

In the dynamic case the user selects the case via TABLE-KEY.

NOTE The PARAMS defined shall be offered to be set by the application.

According to the selection of the TABLE-KEY the corresponding TABLE-STRUCT has to be used by the application to determine the PARAMs to be set by the user.
In the static case that selection is not allowed because TABLE-KEY and TABLE-STRUCT are statically defined by the ODX data.

Drawing: ISO14229-1 WDBI (dynamic)

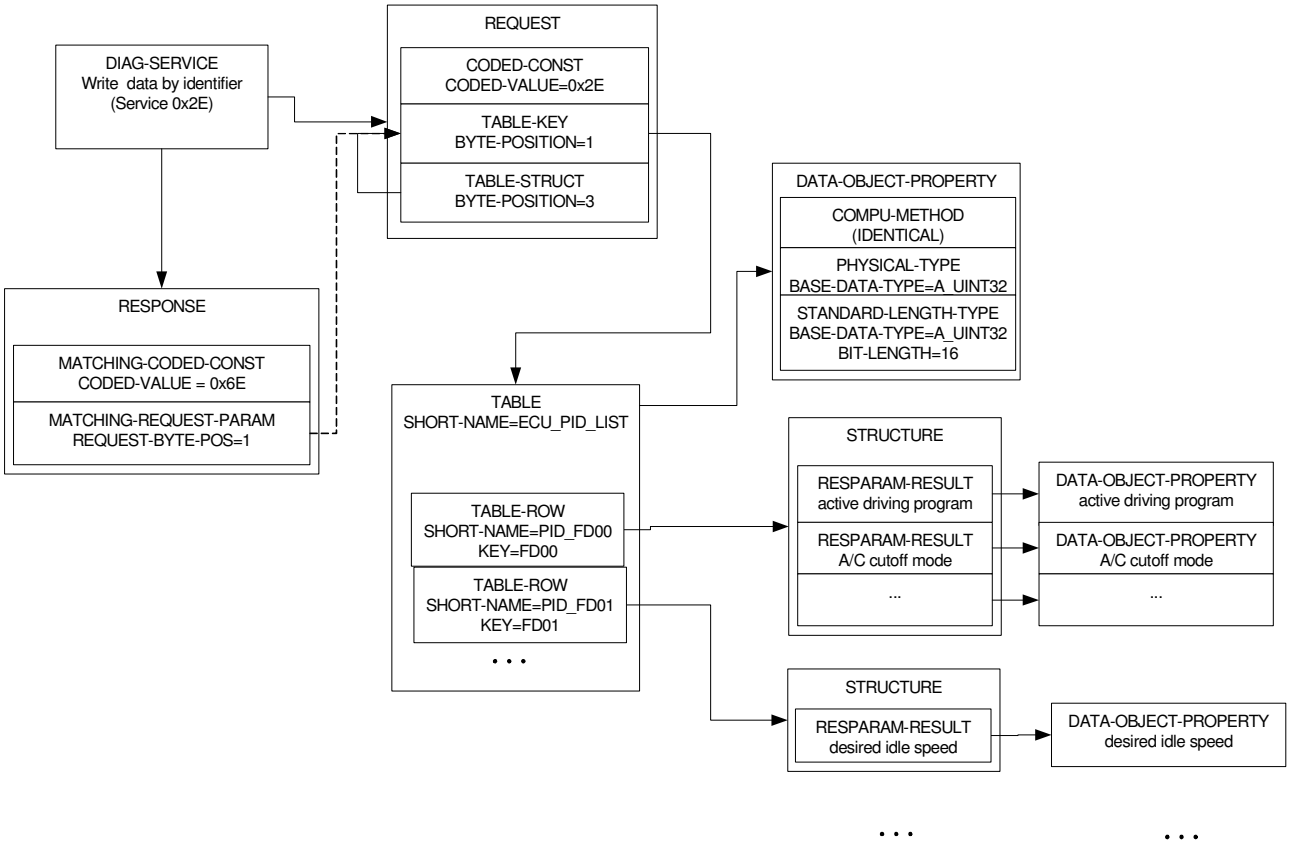


Figure 91 — ISO 14229-1 Write data by identifier (dynamic)

Drawing: ISO14229-1 WDBI (static)

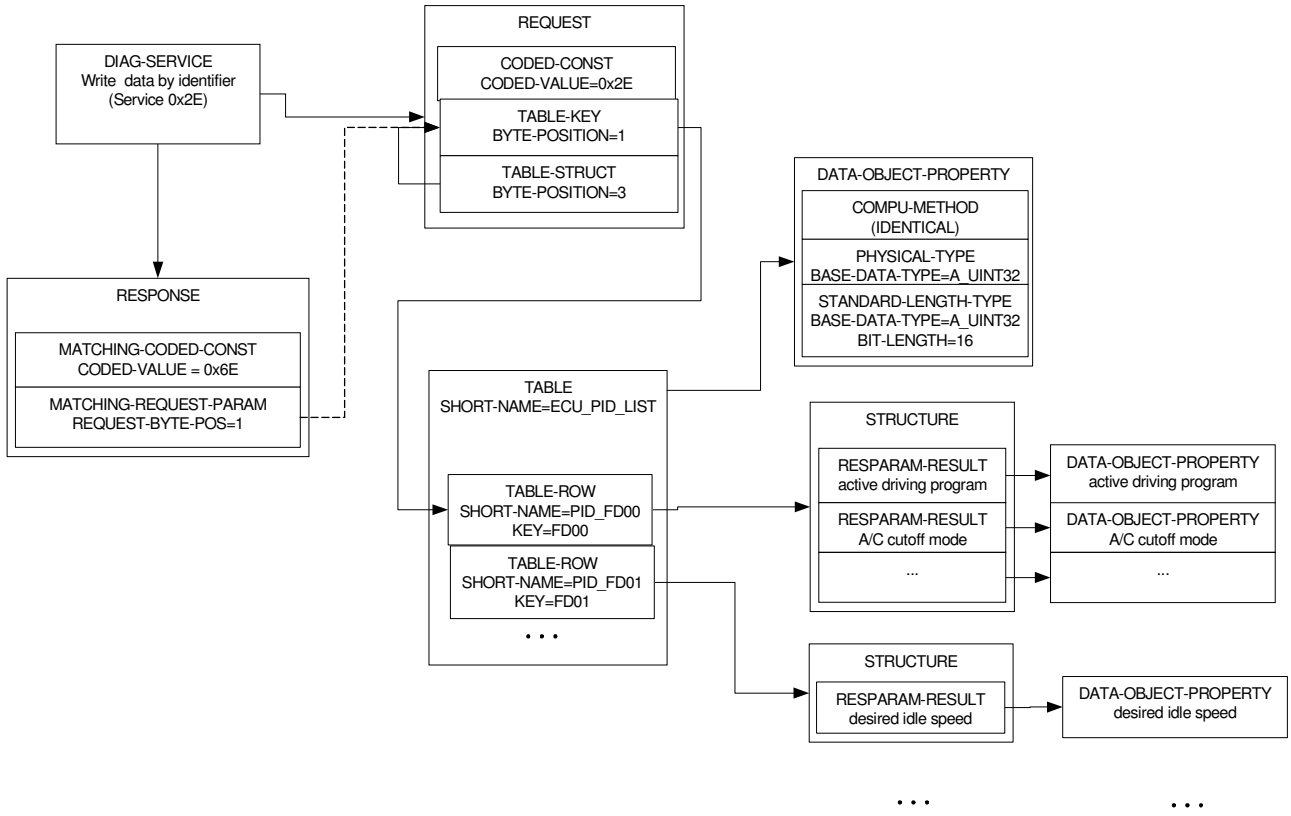


Figure 92 — ISO 14229-1 Write data by identifier (static)

7.3.7 DIAGNOSTIC VARIABLE

In the measurement and calibration (MC) use case the tester has direct access to the ECU memory to read and write data. The ECU provides internal variables to simplify this memory access. An ECU-internal variable maps a logical variable identifier (also called "label") to the memory address of the variable value. This mapping is part of the ECU software; thus, an ECU-internal variable is called "SW-variable".
In the diagnostic use case another communication paradigm is preferred: diagnostic data is typically transferred via diagnostic services or diagnostic jobs that read/write data from/to the ECU. The ECU memory isn't accessed directly.
A runtime system should be capable to utilize SW-variables from the MC use case also in the diagnostic use case. For this reason the notion of diagnostic variables (diag-variables) is introduced in ODX. A diag-variable emulates a SW-variable by calling appropriate diagnostic services for operations like reading or writing a value from/to the ECU.
An ODX conformant diagnostic system allows communicating with an ECU via both diag-variables and diagnostic services. The user can chose the communication paradigm that he is more familiar with.

Drawing: LayerDiagVariable
Package: DiagnosticVariable

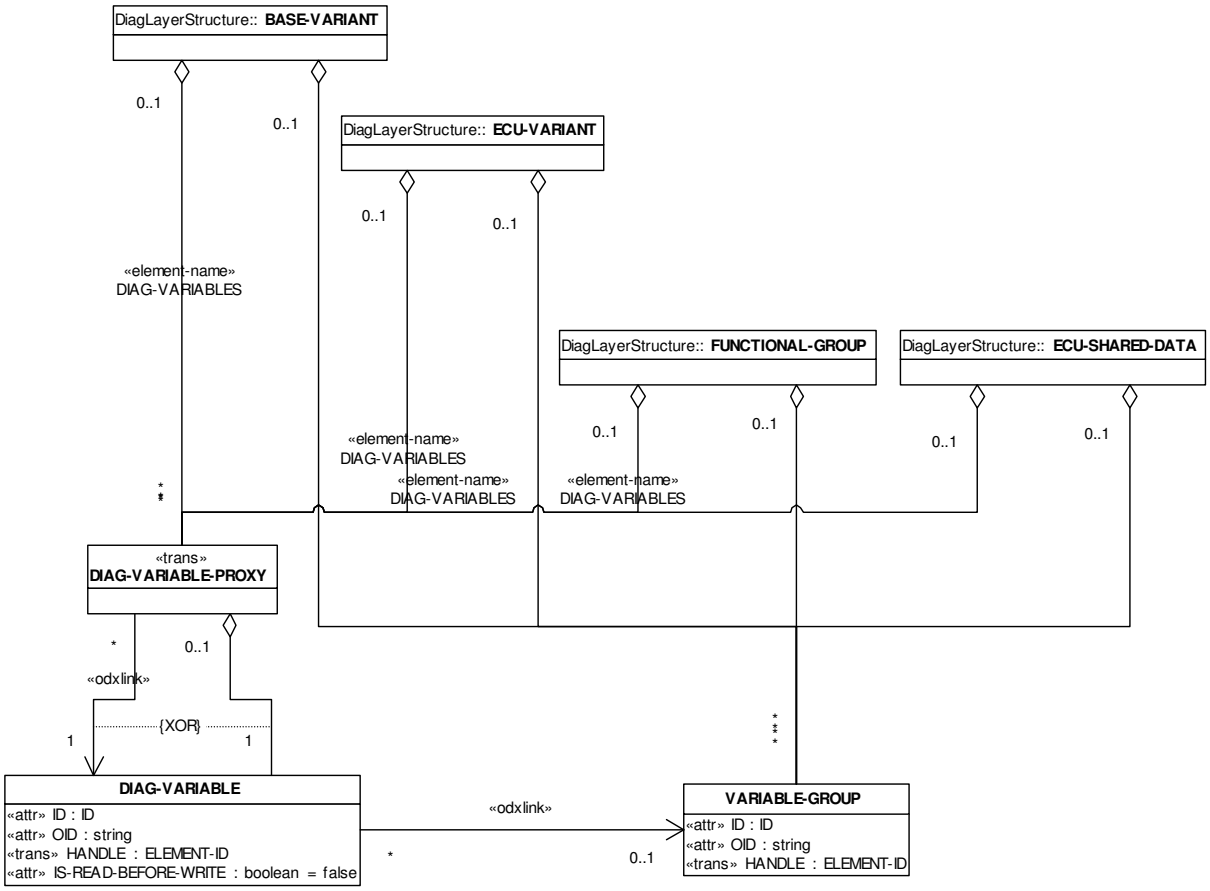


Figure 93 — Diagnostic variable layer

Use cases

— Functional testing of ECUs during development: After setting a value in ECU memory the corresponding diag-variable is read by a diagnostic system while the corresponding SW-variable is read by an MC system. The received values are compared to detect functional errors of the ECU.

— List of all SW-variables which can be accessed via diagnostic services

— Clustering of related services: Diag-variables contain information which services are related (RELATION Element) to each other and deal with the same variable e.g. "read", "write", and "reset to default".

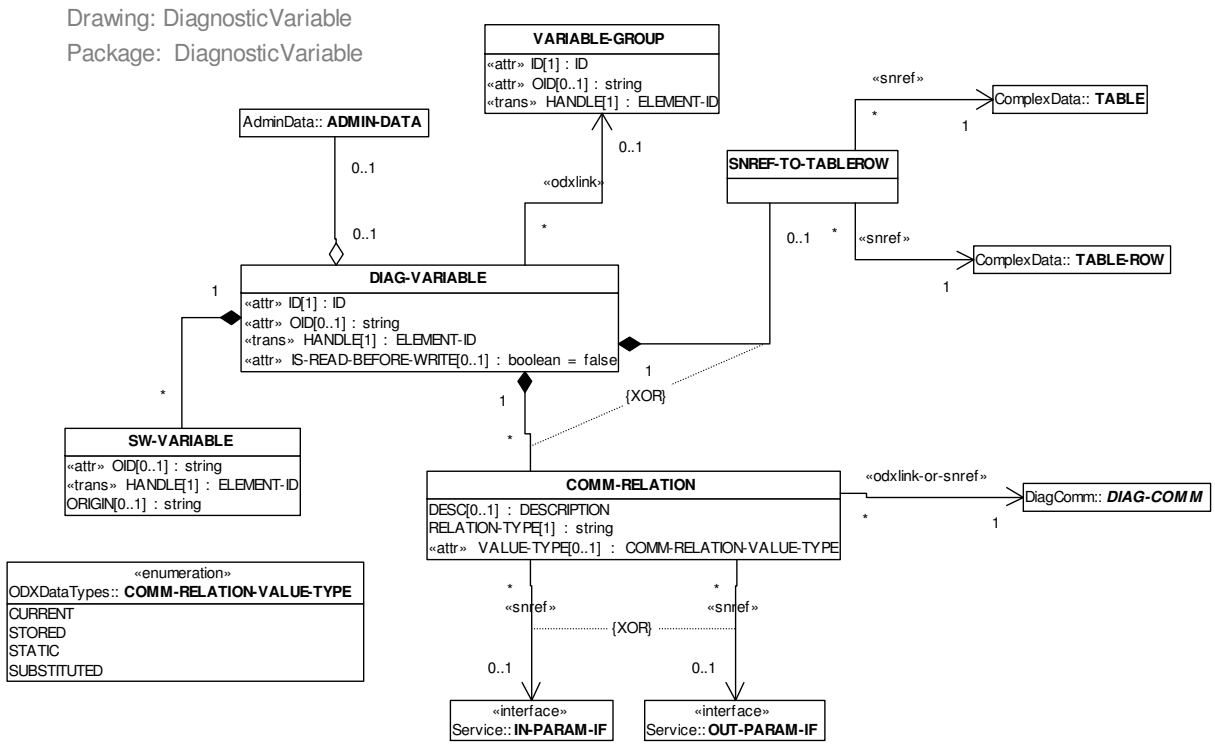


Figure 94 — Diagnostic variable

Structure of DIAG-VARIABLE

A DIAG-VARIABLE typically depends on a number of DIAG-COMM objects (diagnostic services or jobs).
This dependency can be expressed either by placing the appropriate DIAG-COMM objects into the COMM-RELATION element with the adequate value at RELATION (1) or by referencing a TABLE and TABLE-ROW object using short-name references (2).

(1) For example, to read the value of a variable a service is needed that requests the corresponding data from the ECU. A COMM-RELATION object describes the relation between the DIAG-VARIABLE and a DIAG-COMM object; each COMM-RELATION refers to one DIAG-COMM object. Depending on the type of relation there may be a reference to a parameter in the PDU, which contains the variable. The RELATION describes the type of the operation that the referenced DIAG-COMM performs on the diag-variable. Valid values are READ, WRITE, RETURNCONTROL etc. A DIAG-VARIABLE must not have multiple COMM-RELATIONs with the same RELATION string. Modelling the dependency this way provides no possibility to set the corresponding PARAM object (DIAG-SERVICE) and INPUT-PARAM object (SINGLE-ECU-JOB) in order to define e.g. which data identifier to read. This approach leads to inflated ODX data, because for each DIAG-VARIABLE at least one DIAG-COMM object has to be defined with the appropriate value set for the parameter, e.g. by using the PHYSICAL-DEFAULT-VALUE element inside a VALUE parameter.

(2) A second possibility was introduced to describe the dependency between a DIAG-VARIABLE and a DIAG-COMM. Therefore, the new element SNREF-TO-TABLEROW has been added to the ODX model. This new element consists of two

child elements, TABLE-SNREF and TABLE-ROW-SNREF, which reference a TABLE and a TABLE-ROW by SHORT-NAME respectively. This is necessary to target a specific TABLE-ROW unambiguosly. Additionally, the TABLE class has a child element TABLE-DIAG-COMM-CONNECTORS to connect one or more DIAG-COMM objects to a TABLE. The new class references a DIAG-COMM. The corresponding SEMANTIC can be used to define the kind of DIAG-COMM referenced, i.e. a READ and a WRITE service for data identifiers. In DIAG-VARIABLE objects it is only allowed to use either the COMM-RELATION mechanism or the SNREF-TO-TABLEROW mechanism. It is not allowed to use both or none of them.

Operations like reading or writing the value of a diag-variable imply a data transfer from the ECU to the runtime system or vice versa. In this case, , when using the COMM-RELATION element the service parameter that is used for the data transfer is specified by one of the attributes IN-PARAM-IF-SNREF and OUT-PARAM-IF-SNREF. IN-PARAM-IF-SNREF denotes a PARAM of the referenced DIAG-COMM's REQUEST that is used for writing data to the ECU. To read data from the ECU the OUT-PARAM-IF-SNREF is set to the name of the corresponding PARAM of the RESPONSE of the DIAG-COMM. If no data has to be transferred between the runtime system and the ECU no service parameter has to be selected. The runtime system performs such an operation for example when it returns the control of the diag-variable to the ECU.
When using the new SNREF-TO-TABLEROW mechanism, no PARAM-IF-SNREF has to be specified, because by targeting a certain TABLE-ROW element inside a TABLE the DIAG-COMM to use and the PARAM to set are clearly specified:
A runtime system evaluates the referenced TABLE-ROW and by doing this, retrieves the KEY of the TABLE-ROW. The value of the KEY is the value each TABLE-KEY PARAM of the referenced DIAG-COMM’s REQUEST has to be set with.
EXAMPLE Usage of new SNREF-TO-TABLEROW class

```
<BASE-VARIANT ID="BV_ID_1">
       <SHORT-NAME>BV</SHORT-NAME>
       <LONG-NAME>Base Variant</LONG-NAME>
       <DIAG-DATA-DICTIONARY-SPEC>
              <DATA-OBJECT-PROPS>
                     <DATA-OBJECT-PROP ID="DOP_1">
                            <SHORT-NAME>KEYDOP</SHORT-NAME>
                            <COMPU-METHOD>
                                   <CATEGORY>TEXTTABLE</CATEGORY>
                                   <COMPU-INTERNAL-TO-PHYS>
                                          <COMPU-SCALES>
                                                 <COMPU-SCALE>
                                                        <LOWER-LIMIT>65321</LOWER-
LIMIT>
                                                        <UPPER-LIMIT>65321</UPPER-
LIMIT>
                                                        <COMPU-CONST>
                                                               <VT>Engine RPM</VT>
                                                        </COMPU-CONST>
                                                 </COMPU-SCALE>
                                          </COMPU-SCALES>
                                   </COMPU-INTERNAL-TO-PHYS>
                            </COMPU-METHOD>
                            <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                   <BIT-LENGTH>16</BIT-LENGTH>
                            </DIAG-CODED-TYPE>
                            <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                     </DATA-OBJECT-PROP>
                     <DATA-OBJECT-PROP ID="DOP_2">
                            <SHORT-NAME>DOP_2</SHORT-NAME>
                            <COMPU-METHOD>
```

```
                                    <CATEGORY>LINEAR</CATEGORY>
                                    <COMPU-INTERNAL-TO-PHYS>
                                          <COMPU-SCALES>
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>0</LOWER-LIMIT>
                                                      <UPPER-LIMIT>65535</UPPER-
LIMIT>
                                                      <COMPU-RATIONAL-COEFFS>
                                                            <COMPU-NUMERATOR>
                                                                  <V>800</V>
                                                                  <V>0.01</V>
                                                            </COMPU-NUMERATOR>
                                                      </COMPU-RATIONAL-COEFFS>
                                                </COMPU-SCALE>
                                          </COMPU-SCALES>
                                    </COMPU-INTERNAL-TO-PHYS>
                              </COMPU-METHOD>
                              <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                    <BIT-LENGTH>16</BIT-LENGTH>
                              </DIAG-CODED-TYPE>
                              <PHYSICAL-TYPE BASE-DATA-TYPE="A_FLOAT64">
                                    <PRECISION>3</PRECISION>
                              </PHYSICAL-TYPE>
                        </DATA-OBJECT-PROP>
                  </DATA-OBJECT-PROPS>
                  <STRUCTURES>
                        <STRUCTURE ID="STRUCTURE_ID_1">
                              <SHORT-NAME>STRUCTURE_Engine_RPM</SHORT-NAME>
                              <PARAMS>
                                    <PARAM xsi:type="VALUE">
                                          <SHORT-NAME>Engine_RPM</SHORT-NAME>
                                          <BYTE-POSITION>0</BYTE-POSITION>
                                          <DOP-SNREF SHORT-NAME="DOP_2"/>
                                    </PARAM>
                              </PARAMS>
                        </STRUCTURE>
                  </STRUCTURES>
                  <TABLES>
                        <TABLE ID="TABLE_ID_1">
                              <SHORT-NAME>TABLE</SHORT-NAME>
                              <KEY-DOP-REF ID-REF="DOP_1" DOCREF="BV"
DOCTYPE="LAYER"/>
                              <TABLE-ROW ID="TABLE_ROW_ID_1">
                                    <SHORT-NAME>TABLE_ROW</SHORT-NAME>
                                    <KEY>Engine RPM</KEY>
                                    <STRUCTURE-REF ID-REF="STRUCTURE_ID_1"
DOCREF="BV" DOCTYPE="LAYER"/>
                              </TABLE-ROW>
                              <TABLE-DIAG-COMM-CONNECTORS>
                                    <TABLE-DIAG-COMM-CONNECTOR>
                                          <SEMANTIC>READ</SEMANTIC>
                                          <DIAG-COMM-REF ID-REF="DS_1" DOCREF="BV"
DOCTYPE="LAYER"/>
                                    </TABLE-DIAG-COMM-CONNECTOR>
                              </TABLE-DIAG-COMM-CONNECTORS>
                        </TABLE>
                  </TABLES>
            </DIAG-DATA-DICTIONARY-SPEC>
            <DIAG-COMMS>
                  <DIAG-SERVICE ID="DS_1">
                        <SHORT-NAME>RDBI</SHORT-NAME>
                        <REQUEST-REF ID-REF="REQ_1" DOCREF="BV" DOCTYPE="LAYER"/>
                        <POS-RESPONSE-REFS>
                              <POS-RESPONSE-REF ID-REF="POS_RE_1" DOCREF="BV"
DOCTYPE="LAYER"/>
                        </POS-RESPONSE-REFS>
```

```
            </DIAG-SERVICE>
      </DIAG-COMMS>
      <REQUESTS>
            <REQUEST ID="REQUE_1">
                  <SHORT-NAME>REQ_1</SHORT-NAME>
                  <PARAMS>
                        <PARAM xsi:type="CODED-CONST" SEMANTIC="SERVICE-ID">
                              <SHORT-NAME>ServiceID</SHORT-NAME>
                              <BYTE-POSITION>0</BYTE-POSITION>
                              <CODED-VALUE>34</CODED-VALUE>
                              <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE"
BASE-DATA-TYPE="A_UINT32">
                                    <BIT-LENGTH>8</BIT-LENGTH>
                              </DIAG-CODED-TYPE>
                        </PARAM>
                        <PARAM xsi:type="TABLE-KEY" SEMANTIC="LOCAL-ID"
ID="TABLE_KEY_REQUE">
                              <SHORT-NAME>Key</SHORT-NAME>
                              <BYTE-POSITION>1</BYTE-POSITION>
                              <TABLE-REF ID-REF="TABLE_ID_1" DOCREF="BV"
DOCTYPE="LAYER"/>
                        </PARAM>
                  </PARAMS>
            </REQUEST>
      </REQUESTS>
      <POS-RESPONSES>
            <POS-RESPONSE ID="POS_RE_1">
                  <SHORT-NAME>POS_RE_1</SHORT-NAME>
                  <PARAMS>
                        <PARAM xsi:type="CODED-CONST" SEMANTIC="SERVICE-ID">
                              <SHORT-NAME>ServiceID</SHORT-NAME>
                              <BYTE-POSITION>0</BYTE-POSITION>
                              <CODED-VALUE>98</CODED-VALUE>
                              <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE"
BASE-DATA-TYPE="A_UINT32">
                                    <BIT-LENGTH>8</BIT-LENGTH>
                              </DIAG-CODED-TYPE>
                        </PARAM>
                        <PARAM xsi:type="TABLE-KEY" SEMANTIC="LOCAL-ID"
ID="TABLE_KEY_RESPO">
                              <SHORT-NAME>Key</SHORT-NAME>
                              <BYTE-POSITION>1</BYTE-POSITION>
                              <TABLE-REF ID-REF="TABLE_ID_1" DOCREF="BV"
DOCTYPE="LAYER"/>
                        </PARAM>
                        <PARAM xsi:type="TABLE-STRUCT" SEMANTIC="DATA">
                              <SHORT-NAME>Data</SHORT-NAME>
                              <BYTE-POSITION>3</BYTE-POSITION>
                              <TABLE-KEY-REF ID-REF="TABLE_KEY_RESPO"
DOCREF="BV" DOCTYPE="LAYER"/>
                        </PARAM>
                  </PARAMS>
            </POS-RESPONSE>
      </POS-RESPONSES>
      <DIAG-VARIABLES>
            <DIAG-VARIABLE ID="DV_ID_1">
                  <SHORT-NAME>DV</SHORT-NAME>
                  <SNREF-TO-TABLEROW>
                        <TABLE-SNREF SHORT-NAME="TABLE"/>
                        <TABLE-ROW-SNREF SHORT-NAME="TABLE_ROW"/>
                  </SNREF-TO-TABLEROW>
            </DIAG-VARIABLE>
      </DIAG-VARIABLES>
</BASE-VARIANT>
```

The following picture visualizes the SNREF-TO-TABLEROW mechanism and how to use it.

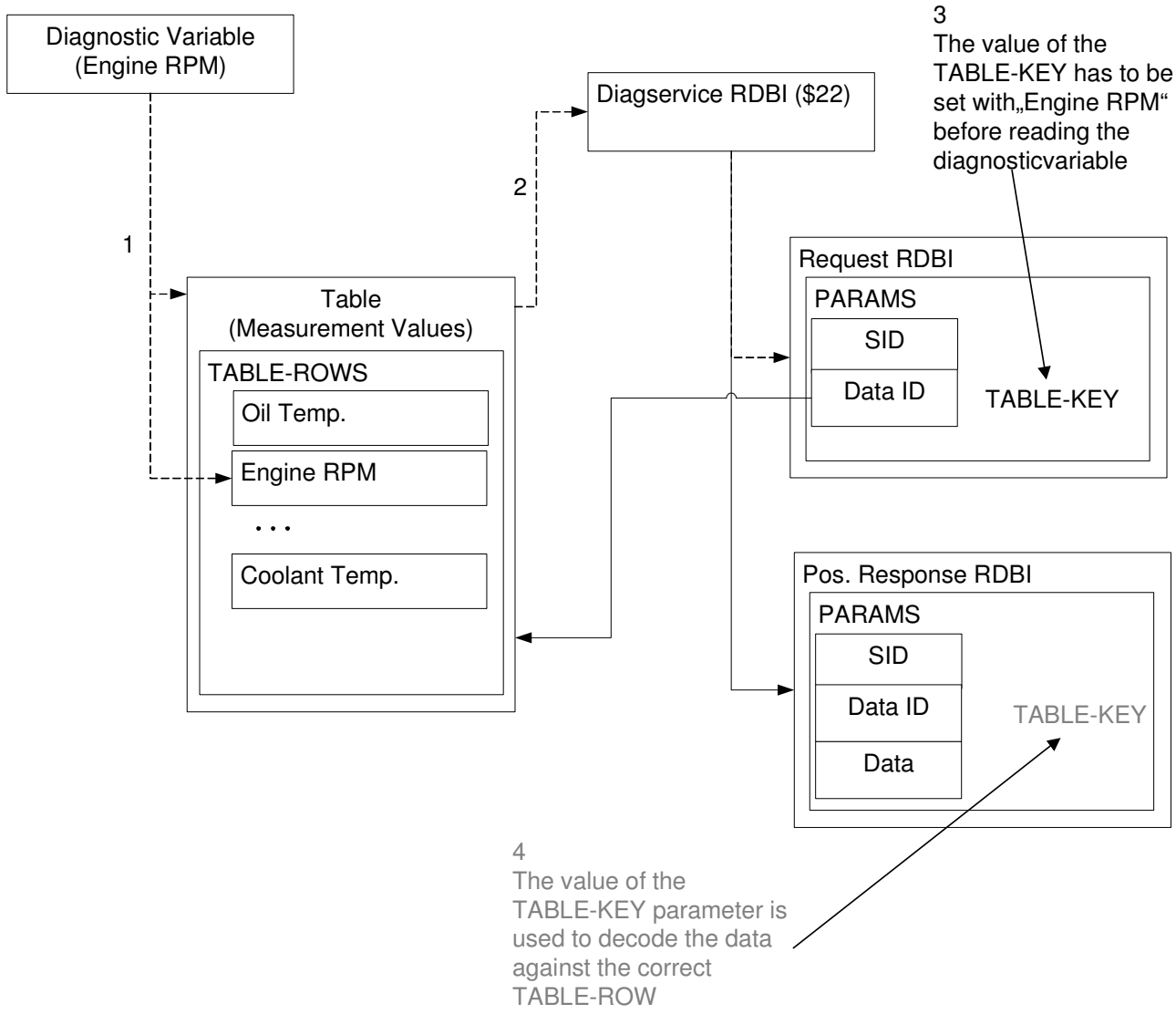


Figure 95 — SNREF-TO-TABLEROW mechanism

If a diag-variable must not be written without reading it before, IS-READ-BEFORE-WRITE attribute must be set to "true". For example, if a diag-variable represent a data set and a single item in this set is to be changed, the whole set must be read and only then the item can be changed and the variable can be written. By default the value "false" is assumed for this attribute.
Typically a diag-variable has a simple data type like integer or string. If the COMM-RELATION mechanism is used, the conversion between the coded and the physical representation for such a diag-variable is done by the DATA-OBJECT-PROP that is associated with the PARAM specified by IN-PARAM-IF-SNREF and OUT-PARAM-IF-SNREF. Both must have identical definitions of PHYSICAL-TYPE. PARAMs with complex DOPs (e.g. STATIC-FIELD or MUX) are not allowed to be referenced.
When using the SNREF-TO-TABLEROW option, the conversion is entirely handled by the specified data related to the referenced TABLE-ROW, e.g. the PARAM objects that are

contained in the referenced STRUCTURE (STRUCTURE-REF in TABLE-ROW) define the conversion between the coded and physical representation.
The VALUE-TYPE defines how the runtime system should handle the transfer of diag-variable values between the ECU and the tester:

- CURRENT: The value should be retrieved from the tester/ECU every time when the diag-variable is accessed. The runtime system sends/receives the actual value. This is the default if no VALUE-TYPE is specified.

- MEASUREMENT: The values describes measurement services, tables etc...

- STORED: When a changed value is retrieved from the tester then the runtime system should store this value for further use. The stored value is sent to the ECU.

- STATIC: The value should be retrieved once from the tester/ECU when the diag-variable is accessed for the first time. After that the value can't be changed anymore. An example for a static value is an ECU-internal identifier.

- SUBSTITUTED: These values are neither current nor stored but values delivered by the ECU as substituted (or intermediate) values.

A SW-VARIABLE object describes a SW-variable that is available as a DIAG-VARIABLE, too. The identifier of the SW-variable specified in the ECU software is used as SHORT-NAME. Additionally, the ORIGIN of the SW-variable, i.e. the ECU software release containing the SW-variable, can be defined.
A DIAG-VARIABLE may be associated with an arbitrary number of SW-VARIABLEs indicating an existing mapping between the represented SW-variables and the diag-variable. One SW-VARIABLE object exists for each SW-variable within specific ECU software that is mapped to the diag-variable. That way, many SW-VARIABLEs may be associated with the same DIAG-VARIABLE. If no SW-VARIABLE is given then the diag-variable is only available in the diagnostic use case.
A DIAG-VARIABLE can refer to a VARIABLE-GROUP. Each VARIABLE-GROUP denotes a category of diag-variables, e.g. measurement variables, calibration parameters, or compound variables. The introduction of VARIABLE-GROUPs is a means to handle large numbers of diag-variables.
EXAMPLE Mapping of two SW-variables (defined by suppliers A and B) to a diag-variable (defined by OEM C)

```
<ECU-VARIANT ID="Engine_ECU_Variant">
       <SHORT-NAME>Engine_ECU_Variant</SHORT-NAME>
       <DIAG-VARIABLES>
              <DIAG-VARIABLE ID="DV_PHY_TMOT" IS-READ-BEFORE-WRITE="false">
                     <SHORT-NAME>DV_PHY_TMOT</SHORT-NAME>
                     <VARIABLE-GROUP-REF ID-REF="VG_MEAS"/>
                     <SW-VARIABLES>
                            <SW-VARIABLE>
                                   <SHORT-NAME>tmot</SHORT-NAME>
                                   <ORIGIN>A</ORIGIN>
                            </SW-VARIABLE>
                            <SW-VARIABLE>
                                   <SHORT-NAME>TMOTK</SHORT-NAME>
                                   <ORIGIN>B</ORIGIN>
                            </SW-VARIABLE>
                     </SW-VARIABLES>
                     <COMM-RELATIONS>
                            <COMM-RELATION VALUE-TYPE="CURRENT">
                                   <RELATION-TYPE>READ</RELATION-TYPE>
                                   <DIAG-COMM-SNREF SHORT-
NAME="DS_ReadMotorTemperature"/>
                                   <OUT-PARAM-IF-SNREF SHORT-NAME="STAT_PHY_TMOT"/>
                            </COMM-RELATION>
```

```
                              <COMM-RELATION VALUE-TYPE="CURRENT">
                                    <RELATION-TYPE>WRITE</RELATION-TYPE>
                                    <DIAG-COMM-SNREF SHORT-
NAME="DS_WriteMotorTemperature"/>
                                    <IN-PARAM-IF-SNREF SHORT-NAME="ARG_PHY_TMOT"/>
                              </COMM-RELATION>
                              <COMM-RELATION VALUE-TYPE="CURRENT">
                                    <RELATION-TYPE>RETURNCONTROL</RELATION-TYPE>
                                    <DIAG-COMM-SNREF SHORT-
NAME="DS_WriteMotorTemperature_END"/>
                              </COMM-RELATION>
                        </COMM-RELATIONS>
                  </DIAG-VARIABLE>
            </DIAG-VARIABLES>
            <VARIABLE-GROUPS>
                  <VARIABLE-GROUP ID="VG_MEAS">
                        <SHORT-NAME>VG_MEAS</SHORT-NAME>
                        <LONG-NAME>measurement variable group</LONG-NAME>
                  </VARIABLE-GROUP>
            </VARIABLE-GROUPS>
</ECU-VARIANT>
```

Inheritance of DIAG-VARIABLEs

DIAG-VARIABLEs are inherited like diagnostic services or jobs; they can be reused or overwritten at lower layers. The inheritance of a diag-variable from the PARENT layer can be eliminated via the NOT-INHERITED-VARIABLE class. For the following usage scenarios we assume that the DIAG-VARIABLE is defined in a BASE-VARIANT instance. If the DIAG-VARIABLE is to be reused in the ECU-VARIANT instance then no further change is necessary. In case that one of the referenced DIAG-COMMs is overwritten in the same ECU-VARIANT instance the association defined by the corresponding COMM-RELATION is overridden; the COMM-RELATION refers to the new DIAG-COMM. The overwriting DIAG-COMM must have identical SHORT-NAMEs of the PARAMS that are used by DIAG-VARIABLE.
The overwriting of a DIAG-VARIABLE is required when a COMM-RELATION or an associated SW-VARIABLE is changed. In this case the complete DIAG-VARIABLE is overwritten including its COMM-RELATIONs. As a consequence, all references to services or TABLEs and to parameters have to be specified again for the ECU-VARIANT even if they haven't changed from the BASE-VARIANT.

7.3.8 DYNAMICALLY DEFINED MESSAGES

Drawing: DynDefined
Package: DynDefined

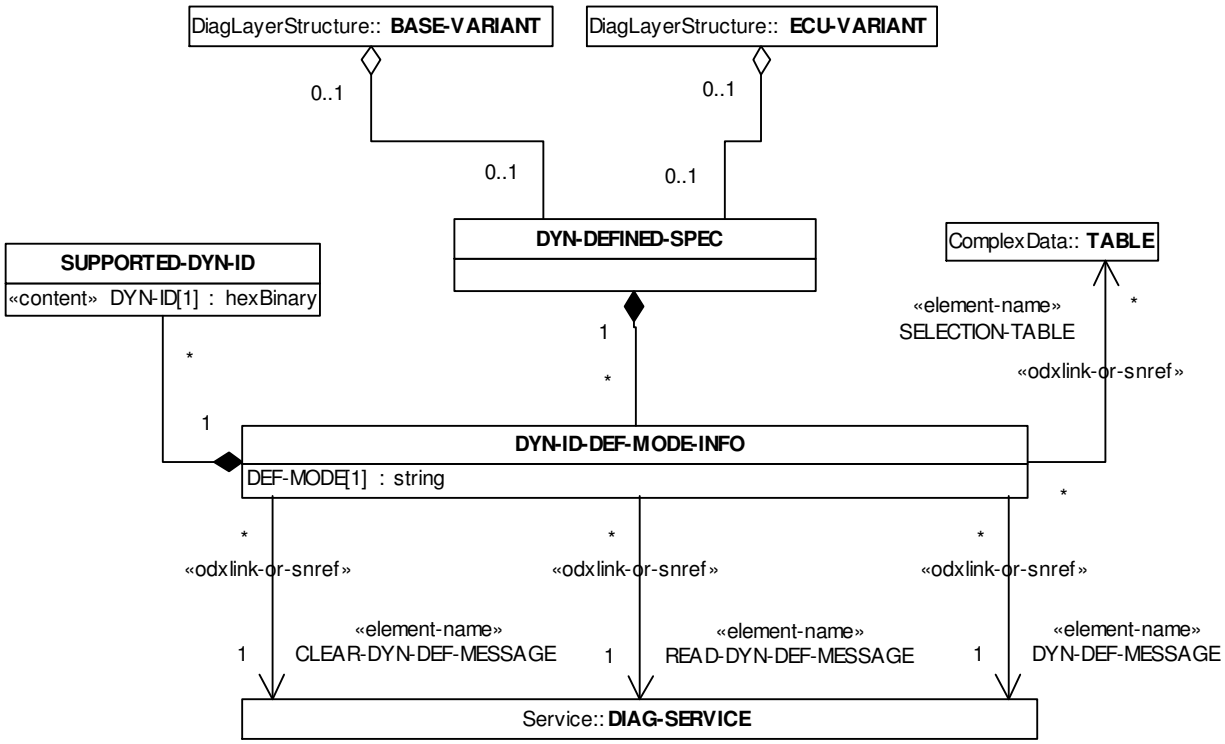


Figure 96 — DYN-DEFINED-SPEC

With some protocols (e.g. ISO 14230 or ISO 14229-1) it is possible to dynamically define data records at run time, which contain values from other records identified by a local identifier (LOCAL-ID), a common identifier (COMMON-ID) or an address in the ECU memory, etc. The new data records are identified by a DYN-ID (dynamically defined identifier). The object DYN-DEFINED-SPEC lists all supported dynamically defined identifiers of the ECU (SUPPORTED-DYN-ID) and references all TABLEs, which contain information about existing data records and can be used for creation of new DYN-ID records. The supported DYN-IDs and the TABLEs are grouped by the definition mode. Therefore the class DYN-ID-DEF-MODE-INFO was introduced as a wrapper for one group of DYN-IDs and the appropriate TABLEs. Its member DEF-MODE indicates the definition mode: by LOCAL-ID, COMMON-ID, etc. Each instance of DYN-ID-DEF-MODE-INFO refers to three DIAG-COMMs used to define new DYN-IDs, to read data using a DYN-ID and to clear a DYN-ID defined before. The DIAGNOSTIC-CLASS of these DIAG-COMMs must have the values DYN-DEF-MESSAGE, READ-DYN-DEF-MESSAGE or CLEAR-DYN-DEF-MESSAGE respectively. Within one DYN-DEFINED-SPEC there should not exist multiple DYN-ID-DEF-MODE-INFO instances with the same DEF-MODE value.

Within a diagnostic layer the DYN-DEFINED-SPEC must be defined, if this layer is used for communication with the ECU and the DIAG-COMMs with DIAGNOSTIC-CLASS="DYN-DEF-MESSAGE", "READ-DYN-DEF-MESSAGE" and "CLEAR-DYN-DEF-MESSAGE" are defined and to be used. All TABLEs used for creation of new DYN-ID records must be referenced from this DYN-DEFINED-SPEC. If an ECU-VARIANT does not have an own DYN-DEFINED-SPEC, the DYN-DEFINED-SPEC of the parent BASE-VARIANT is valid. If the ECU-VARIANT defines its own DYN-DEFINED-SPEC, it completly replaces the one of the BASE-VARIANT. See section 7.4.2 to get an example for use of DYN-DEFINED-SPEC. The definition of an empty DYN-DEFINED-SPEC at the ECU-VARIANT means that this ECU-VARIANT doesn't support the mechanism of the DYN-DEFINED-SPEC independent of the BASE-VARIANT.

7.3.9 SESSION AND SECURITY HANDLING

The session and security submodel covers two aspects: The first is to describe possible state transitions resulting from the execution of a DIAG-COMM. The second is to describe preconditions for the execution of a DIAG-COMM. Figure °86 shows the data model for the two features.

Drawing: SessionSecurity
Package: SessionSecurity

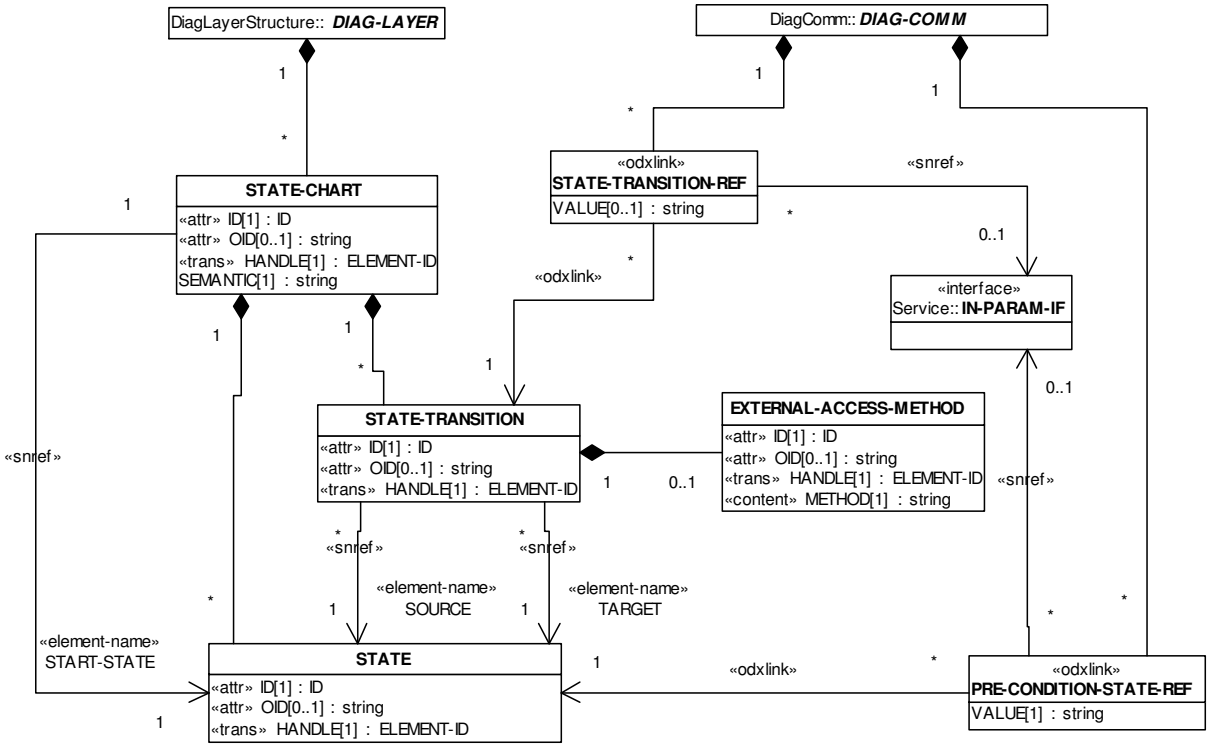


Figure 97 — STATE-CHART

Both the session and the security handling are modelled within one state machine model. The element STATE-CHART stores the possible STATEs of an ECU. SEMANTIC defines the type of the STATE-CHART. It has the predefined values "SESSION" and "SECURITY". SEMANTIC is extendible by the user. The start state of the state machine

is defined by the START-STATE snref at STATE-CHART. A state transition within one state machine is modelled by the STATE-TRANSITION element at STATE-CHART. Each STATE-TRANSITION references one source (SOURCE) and one target state (TARGET). A STATE-TRANSITION can have an optional reference to an EXTERNAL-ACCESS-METHOD.
The EXTERNAL-ACCESS-METHOD allows to specify a OEM-specific method necessary to perform the STATE-TRANSITION. For security-critical applications this link can be used. The attribute METHOD specifies which method is to be used. The values or METHOD need to be defined OEM-specifically and the semantics of every method must be implemented into an OEM-specific extension of a MVCI server
STATE-TRANSITION-REFs and PRE-CONDITION-STATE-REFs can be used at DIAG-COMMs. If STATE-TRANSITION-REF is used, this has the following semantic: If the DIAG-COMM is successfully executed, the state of the ECU changes from the specified SOURCE to the specified TARGET state. A DIAG-COMM is executable in all SOURCE states defined in its referenced STATE-TRANSITIONs. Self transitions can be described by STATE-TRANSITIONs with identical SOURCE and TARGET states.
PRE-CONDITION-STATE-REF can be used to define the allowed states for the execution of the DIAG-COMM in cases where the execution of the DIAG-COMM does not result in a state-transition of the ECU but its execution is bound to the condition that the ECU already is in a certain state. If a STATE-TRANSITION-REF is used, there may but does not have to be a PRE-CONDITION-STATE-REF for the source states of the referenced transitions. If neither STATE-TRANSITION-REF nor PRE-CONDITION-STATE-REF are used, the DIAG-COMM is executable in all states and the execution does not result in a state transition. If STATE-TRANSITION-REFs and/or PRE-CONDITION-STATE-REFs are used, the DIAG-COMM is executable in the SOURCE states of the referenced STATE-TRANSITIONs and in the referenced preconditions' STATEs but no other states.
The optional VALUE together with the also optional IN-PARAM-IF snref at STATE-TRANSITION-REF and PRE-CONDITION-STATE-REF can be used if the STATE-TRANSITIONs and pre-condition STATEs are dependent on the values of the referenced PARAMs.
The following constraints (Checker rules) hold for the data model:

1. Within a STATE-TRANSITION it is only allowed to reference STATEs that are contained in the same STATE-CHART as the STATE-TRANSITION itself.

2. Only the following IN-PARAM-IF subclasses are allowed within STATE-TRANSITION-REF/STATE-REF: VALUE, CODED-CONST, PHYS-CONST, TABLE-KEY.

3. If the IN-PARAM-IF referenced by STATE-TRANSITION-REF or PRE-CONDITION-STATE-REF targets a TABLE-KEY, this TABLE-KEY must have an odx-link to a TABLE-ROW, not a TABLE.

4. Only one STATE-CHART per CATEGORY is allowed within one DIAG-LAYER.

5. If STATE-TRANSITION-REF.IN-PARAM-IF is present STATE-TRANSITION-REF.VALUE must be defined.

6. Only DATA-OBJECT-PROPs may be referenced by IN-PARAM-IFs within STATE-TRANSITION/STATE-REF.

7.3.10 VEHICLE INFORMATION

7.3.10.1 VEHICLE IDENTIFICATION DATA

The vehicle information specification serves two purposes: One purpose is the identification of a vehicle through the fixing of a set of values. The other purpose is to give a runtime sytem (e.g. MCD 3D / MVCI Diagnostic Server API) access to the vehicle by describing the vehicle topology.
The data model as shown in Figure 98 covers the first aspect. The data model as shown in Figure 99 covers the second aspect.

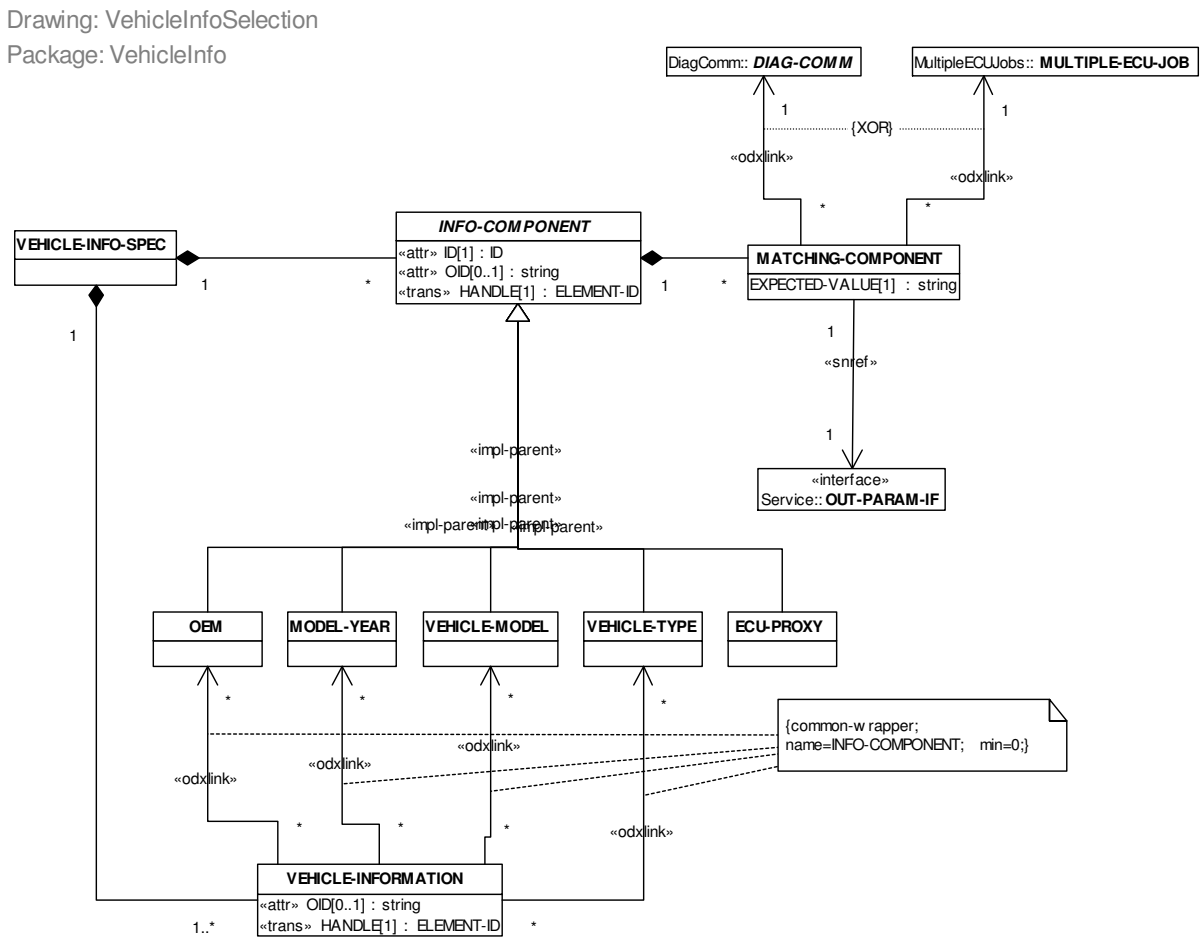


Figure 98 — Vehicle information selection

The vehicle identification data model consists of a set of INFO-COMPONENTs. An INFO-COMPONENT describes one aspect of a vehicle that narrows down the set of possible vehicle topologies (VEHICLE-INFORMATION). Currently, the following INFO-COMPONENTS have been modelled: the OEM (e.g. “Volkswagen”), the MODEL-YEAR (e.g. “2003”), the VEHICLE-MODEL (e.g. “Golf V”) and the VEHICLE-TYPE (e.g. “passenger car”). A special kind of INFO-COMPONENT is the so-called ECU-PROXY. It is used to establish an alternative description for a family of built-in ECUs for external presentation. A vehicle may have multiple possible engines with different engine control

ECUs. Within the BASE-VARIANTs only internal names for these ECUs are used (e.g. ZXH20). If, for vehicle identification purposes, it is necessary for a service technician to enter some information about the engine manually, because it cannot be read out automatically, he will only be able to enter the externally known name of the engine (e.g. “1.9 TDI”). This is made possible through an ECU-PROXY.
If all INFO-COMPONENTS are identified (a value has been assigned to them), ideally one or maybe a small set of possible vehicle topology descriptions (instances of VEHICLE-INFORMATION) has been determined. This is modelled through the references from the VEHICLE-INFORMATION class to the different INFO-COMPONENTs. ECU-PROXYs are referenced indirectly through the LOGICAL-LINKs contained within a VEHICLE-INFORMATION specification.
The values for the INFO-COMPONENTs may be established in two different ways: by interactive data entry by e.g. the service technician or by communication with the vehicle. For the latter case, every INFO-COMPONENT may contain an arbitrary number of MATCHING-COMPONENTs. These contain references to output or response parameters of jobs or diagnostic services, respectively. An expected value is specified for every MATCHING-COMPONENT. If this expected value is matched by the output or response parameter at runtime, a specific INFO-COMPONENT is found. Between the different MATCHING-COMPONENTs of one INFO-COMPONENT a logical AND-constraint is assumed. This means a specific INFO-COMPONENT is established, if all contained MATCHING-COMPONENTs actually matched.

7.3.10.2 VEHICLE TOPOLOGY

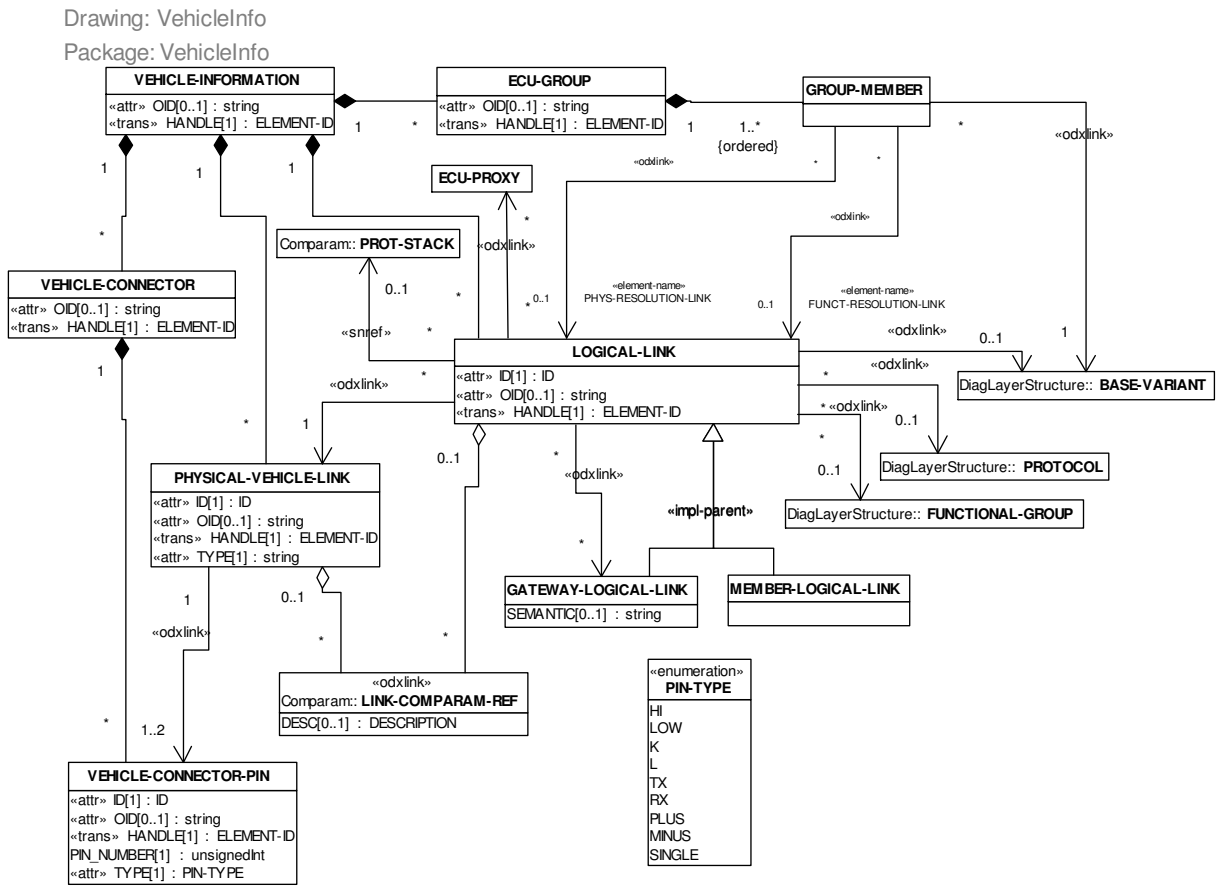


Figure 99 — Vehicle information

The vehicle topology data model's main element is the VEHICLE-INFORMATION. One instance of VEHICLE-INFORMATION describes the vehicle topology of one designated set of vehicles, e.g. all vehicles of one model within one model year. A VEHICLE-INFORMATION contains information about the VEHICLE-CONNECTOR, the available PHYSICAL-VEHICLE-LINKs and the available LOGICAL-LINKs.
A VEHICLE-CONNECTOR contains a description of its pins (VEHICLE-CONNECTOR-PIN). Every pin has a number (PIN-NUMBER). This number must be unique within one VEHICLE-CONNECTOR. A PHYSICAL-VEHICLE-LINK has a TYPE, which describes the physical communication layer usable on that link (e.g. K-LINE or CAN) and identifies the VEHICLE-CONNECTOR-PIN that has to be used in order to communicate via this PHYSICAL-VEHICLE-LINK. The value of the TYPE of the PHYSICAL-VEHICLE-LINK must exist within the list of POSSIBLE-PHYSICAL-LINK-TYPEs of the used PROTOCOL.
A LOGICAL-LINK is the representation of one access path to a protocol usable for ECU communication, a functional group of ECUs or an ECU base variant. Multiple references to DIAG-LAYERs must be used to specify the complete access information to an ECU. As it is possible for a BASE-VARIANT to implement multiple protocols, one PROTOCOL must explicitly be referenced by a logical link to determine which protocol is to be used on this access path to the ECU. A LOGICAL-LINK can either reference a BASE-VARIANT alone or a PROTOCOL alone or a PROTOCOL and a BASE-VARIANT or a PROTOCOL and a FUNCTIONAL-GROUP. A reference to a BASE-VARIANT and PROTOCOL has the

implicit meaning, that all FUNCTIONAL-GROUPS that inherit from that PROTOCOL and to that BASE-VARIANT are valid.
Two subclasses of LOGICAL-LINK exist. One subclass to identify a logical link to a gateway ECU (GATEWAY-LOGICAL-LINK) and one subclass to identify a regular ECU that may be accessed either directly or through a gateway (MEMBER-LOGICAL-LINK). Every logical link may reference other logical links. The referenced target identifies a gateway through which the ECU represented by the reference source can be accessed. By letting a GATEWAY-LOGICAL-LINK reference another GATEWAY-LOGICAL-LINK, a hierarchy of gateways can be specified.
The optional reference to a PROT-STACK indicates which set fo protocol specific communication parameters are used for a certain LOGICAL-LINK. LOGICAL-LINKs and PHYSICAL-VEHICLE-LINKs may both reference COM-PARAMs. COM-PARAMs at the physical vehicle link describe baud rates, timings etc. that are dependent on the physical capabilities of the hardware link (e.g. the CAN bus or the K-Line). COM-PARAMs at the logical link are used to define communication parameters dependent on the built-in of an ECU into the vehicle (e.g. behind a gateway or not).
An ECU-GROUP allows grouping of different BASE-VARIANTs together that are to be seen as alternatives within the VEHICLE-INFORMATION. Consider a VEHICLE-INFORMATION element describing a complete vehicle model and this vehicle model has an alternative built-in of ECUs serving the same purpose (e.g. multiple different radios). These ECUs even can have the same diagnostic address. In this case a separate logical link for each one of these radios would be contained in the VEHICLE-INFORMATION. However, since only one of the radios can actually occur in a given vehicle, all these logical links are grouped in an ECU-GROUP. Every GROUP-MEMBER of an ECU-GROUP therefore references a BASE-VARIANT and a LOGICAL-LINK since the same BASE-VARIANT may be accessible through multiple LOGICAL-LINKs. Every ECU-GROUP within one VEHICLE-INFO-SPEC needs to be resolved, before functional communication can consider overridden response messages. The references from GROUP-MEMBER to LOGICAL-LINK named PHYS-RESOLUTION-LINK-REF and FUNCT-LOGICAL-LINK-REF indicate whether this resolution is done by physical or functional addressing. For base variant identification the MATCHING-PARAMETERs of the BASE-VARIANT-PATTERNs at each corresponding BASE-VARIANT are used (see also the chapter about base variant identification). Either a PHYS-RESOLUTION-LINK or a FUNCT-RESOLUTION-LINK must exist for every GROUP-MEMBER. The FUNCT-RESOLUTION-LINK points to a LOGICAL-LINK that references a FUNCTIONAL-GROUP and no BASE-VARIANT. The PHYS-RESOLUTION-LINK points to a LOGICAL-LINK that references a BASE-VARIANT.

Runtime Behaviour

Currently, the MCD 3D / MVCI Diagnostic Server API is only able to use the vehicle topology data and not the vehicle identification data. Communication with ECUs within the vehicle is established by opening a logical link. The data of the location (e.g. a base variant) referenced by the logical link is then loaded into the system and the location's defined services and jobs can be executed.

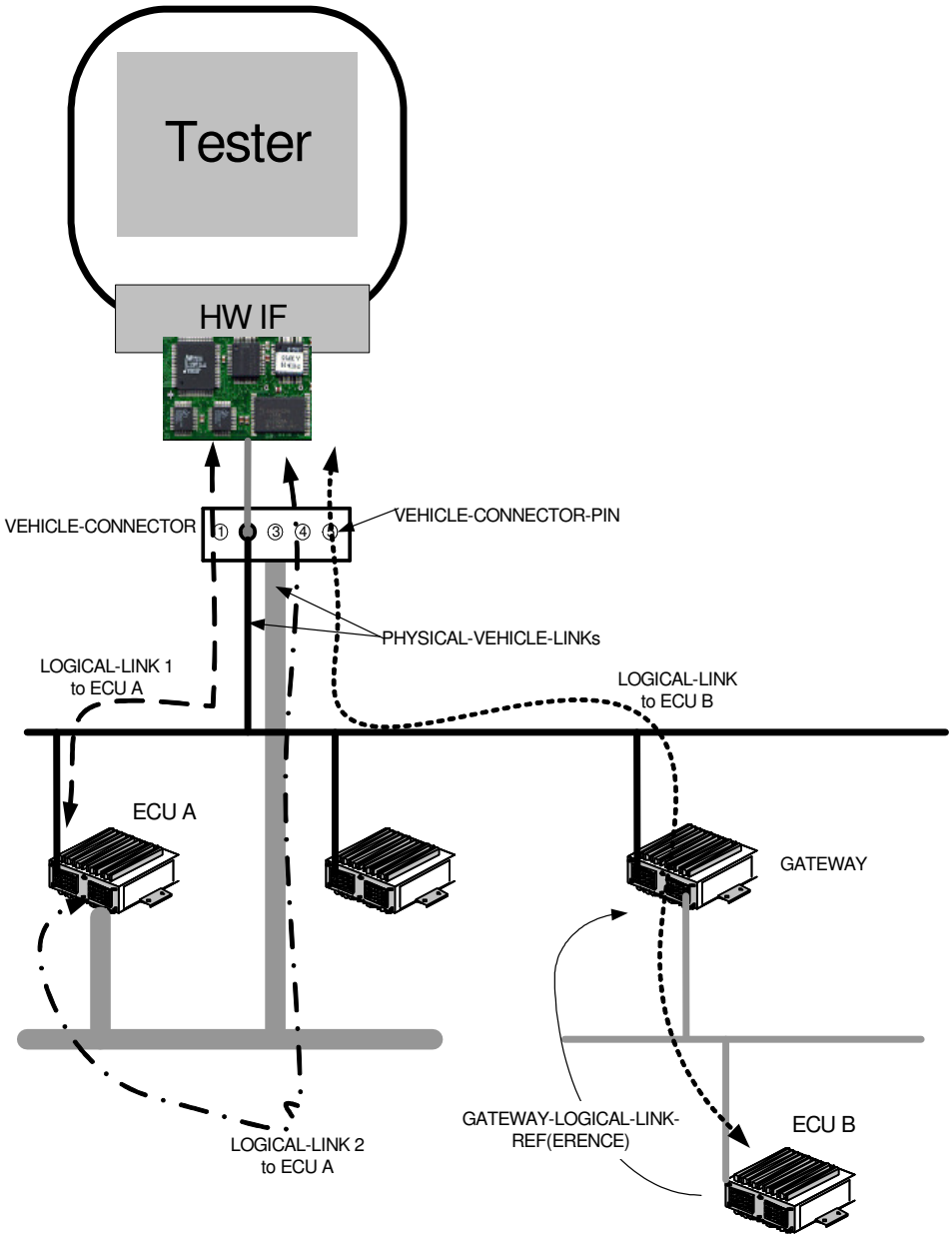


Figure 100 — Vehicle topology example

7.3.11 MULTIPLE ECU JOBS

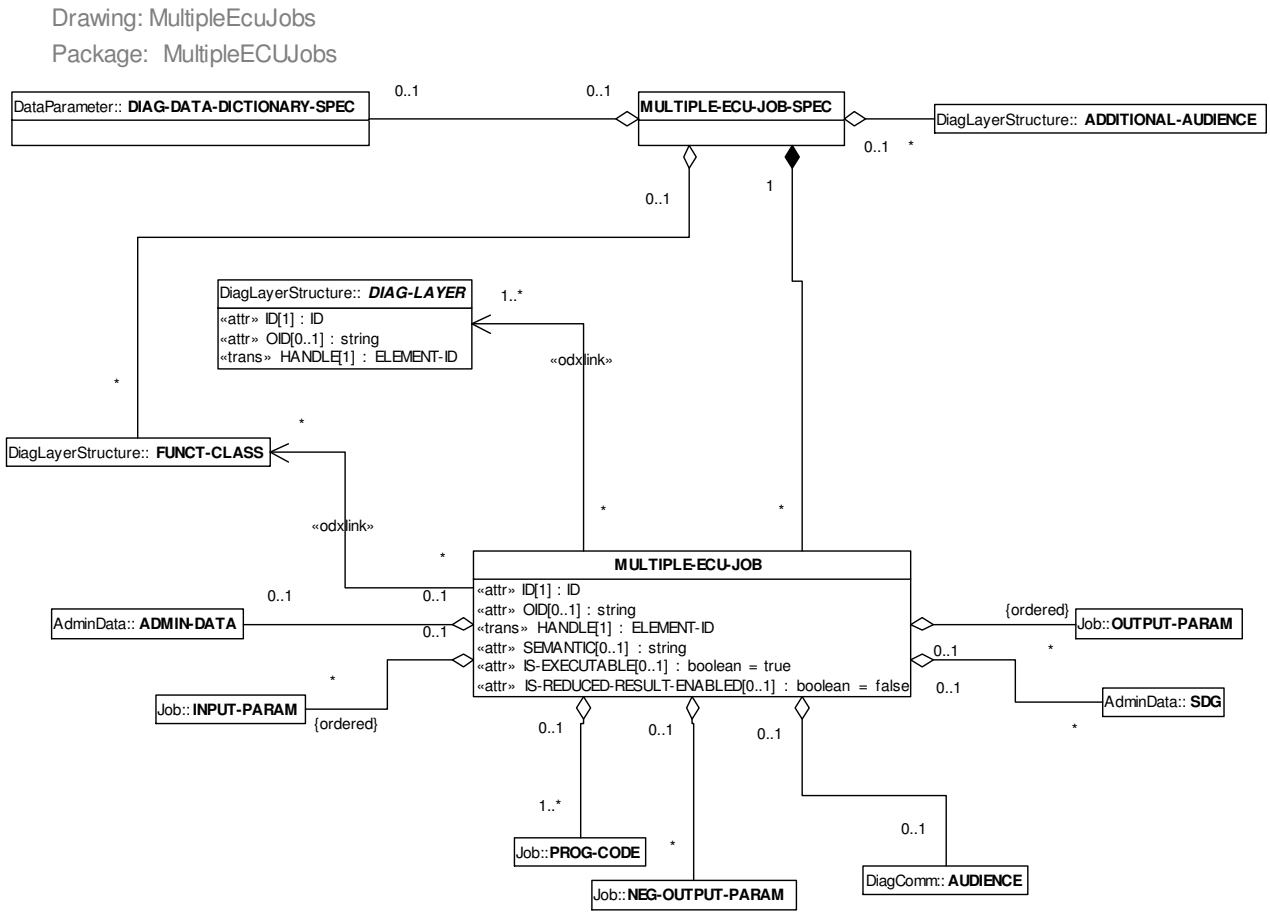


Figure 101 — Multiple ECU jobs

The MULTIPLE-ECU-JOB-SPEC is one kind of ODX-CATEGORY. An arbitrary number of such jobs may be contained in one MULTIPLE-ECU-JOB-SPEC.
In contrast to SINGLE-ECU-JOBS, the code of MULTIPLE-ECU-JOBs contains calls to services (and jobs) of more than one ECU. By consequence, a MULTIPLE-ECU-JOB cannot be assigned to the diagnostic data specification of one dedicated ECU (e.g. a BASE-VARIANT). At execution time, a MULTIPLE-ECU-JOB is not bound to one logical link (as a SINGLE-ECU-JOB is) within the MCD 3D / MVCI server. Rather, a MULTIPLE-ECU-JOB may open and close logical links deliberately and communicate to multiple ECUs simultaneously. An application domain for a MULTIPLE-ECU-JOB is the identification of a vehicle, where the job communicates to multiple ECU base variants to identify the vehicle model, make and year.
Multiple ECU jobs are an alternative way to communicate with multiple ECUs next to functional services. However, they do not use functional addressing.
As MULTIPLE-ECU-JOBs cannot be assigned to one ECU, they cannot be included in a DIAG-LAYER specification. In fact, they are a separate ODX-CATEGORY. In addition to ADMIN-DATA and COMPANY-DATA, which are inherited from ODX-CATEGORY, the MULTIPLE-ECU-JOB-SPEC class also has an aggregation to a DIAG-DATA-DICTIONARY-SPEC. This is needed to specify the data structures for the input and output

parameter structures of the contained MULTIPLE-ECU-JOBs. Furthermore, new functional classes (FUNCT-CLASS) can be defined within a MULTIPLE-ECU-JOB-SPEC. The class MULTIPLE-ECU-JOB is similar to a SINGLE-ECU-JOB and thus aggregates ADMIN-DATA, INPUT- and OUTPUT-PARAMs as well as NEG-OUTPUT-PARAMs, AUDIENCE and SDG classes. A MULTIPLE-ECU-JOB may also reference a set of functional classes (FUNCT-CLASS). However, a MULTIPLE-ECU-JOB is extended with an association to one or multiple DIAG-LAYERs. Within an instance, the references to DIAG-LAYERs are used to specify data of which DIAG-LAYERs is needed for the execution of the MULTIPLE-ECU-JOB. Thus, if a MULTIPLE-ECU-JOB's code calls a diagnostic service of a BASE-VARIANT called Engine_Control_Module, the data specification for this MULTIPLE-ECU-JOB shall contain a reference to the Engine_Control_Module BASE-VARIANT diagnostic layer. The consistency between the code of the MULTIPLE-ECU-JOB and the specified parameter and DIAG-LAYER reference data must be maintained manually or by a separate analysis tool. A standard-conformant tool does not have to check this consistency.

EXAMPLE The following example shows the structure of a MULTIPLE-ECU-JOB-SPEC diagnostic data specification. The only contained MULTIPLE-ECU-JOB returns a vehicle's model, make and year. To perform this vehicle identification, the job needs to retrieve data from two ECUs. Their respective base variant data specifications are referenced within the DIAG-LAYER-REFs part. As vehicle identification is a task mainly performed at the dealerships and service bays, the audience is restricted to aftermarket and aftersales.

```
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation=" odx.xsd">
      <MULTIPLE-ECU-JOB-SPEC ID="ID_73103450">
            <SHORT-NAME>MEJS</SHORT-NAME>
            <MULTIPLE-ECU-JOBS>
                  <MULTIPLE-ECU-JOB ID="ID_73101" SEMANTIC="IDENTIFICATION">
                        <SHORT-NAME>Vehicle_Identification</SHORT-NAME>
                        <PROG-CODES>
                              <PROG-CODE>
                                    <CODE-FILE>Vehicle_Ident</CODE-FILE>
                                    <SYNTAX>CLASS</SYNTAX>
                                    <REVISION>1.3.4</REVISION>
                              </PROG-CODE>
                        </PROG-CODES>
                        <OUTPUT-PARAMS>
                              <OUTPUT-PARAM ID="ID_73107">
                                    <SHORT-NAME>Vehicle_Model</SHORT-NAME>
                                    <LONG-NAME>Name of vehicle model</LONG-
NAME>
                                    <DOP-BASE-REF ID-REF="ID_73102"/>
                              </OUTPUT-PARAM>
                              <OUTPUT-PARAM ID="ID_73108">
                                    <SHORT-NAME>Vehicle_Manufacturer</SHORT-
NAME>
                                    <LONG-NAME>Name of vehicle
manufacturer</LONG-NAME>
                                    <DOP-BASE-REF ID-REF="ID_73103"/>
                              </OUTPUT-PARAM>
                              <OUTPUT-PARAM ID="ID_73109">
                                    <SHORT-NAME>Vehicle_Year</SHORT-NAME>
                                    <LONG-NAME>Year vehicle has been
manufactured</LONG-NAME>
                                    <DOP-BASE-REF ID-REF="ID_73104"/>
                              </OUTPUT-PARAM>
                        </OUTPUT-PARAMS>
                        <DIAG-LAYER-REFS>
    <DIAG-LAYER-REF ID-REF="ID_73105" DOCREF="ECM" DOCTYPE="CONTAINER"
    REVISION="2.1.3"/>
    <DIAG-LAYER-REF ID-REF="ID_73106" DOCREF="BCM" DOCTYPE="LAYER"
REVISION="4.1.5"/>
                        </DIAG-LAYER-REFS>
                        <AUDIENCE IS-AFTERMARKET="true" IS-AFTERSALES="true"/>
```

```
                    </MULTIPLE-ECU-JOB>
                </MULTIPLE-ECU-JOBS>
            </MULTIPLE-ECU-JOB-SPEC>
</ODX>
```

7.3.12 DATA TYPES

There are a few type definitions that are used throughout the data model to ensure consistent structures.

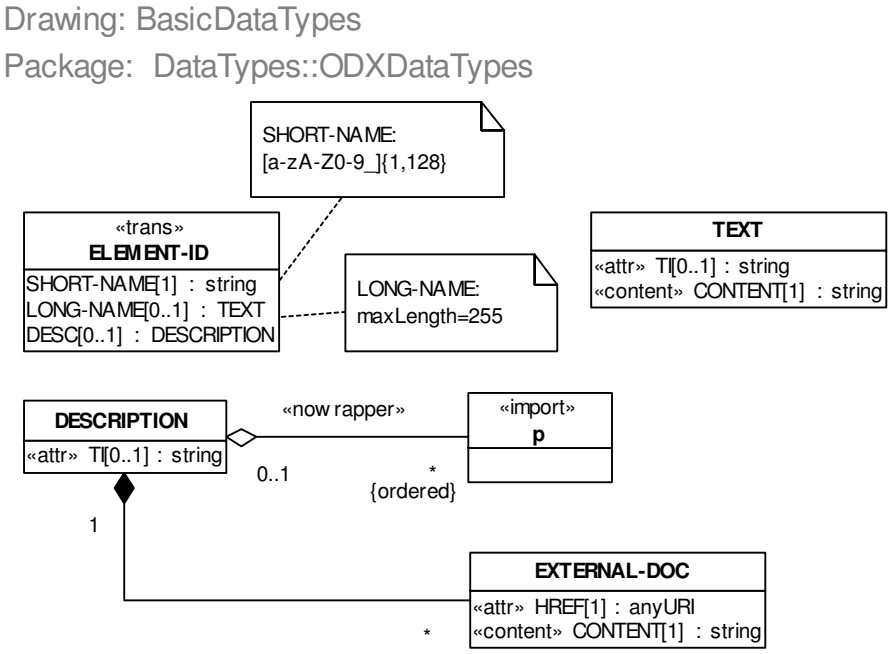


Figure 102 — Basic data types

— The data type ELEMENT-ID is a representation of the three elements SHORT-NAME, LONG-NAME and DESC (see also 7.1.1).

— The DESCRIPTION data type is used for all DESC elements and provides a method to add formatted documentation to the corresponding data element. The sub-element EXTERNAL-DOC can be used to refer to any kind of external document. With the HREF attribute the document is addressed and the CONTENT may provide a description or meta information. The external document can be a file or anything else. How to resolve this element is not specified by the document. It can be done in a system specific manner and is not a mandatory feature of an ODX compliant tool.

— The data type p represents the <p> tag from XHTML. Only a subset of what's defined with the XHTML recommendation can be used to capture formatted documentation.

— The TEXT data type is being used at all places where an identifier needs to be assigned to a string e.g. to support internationalisation.

— The EMPTY data type is being used to denote an empty content model within the XML implementation.

7.3.13 REFERENCES

7.3.13.1 OVERVIEW

Some parts of an ODX document like UNITs or DOPs may be needed more than once. Therefore they can be reused by reference. The referenced part is called "link target"; the ODX element that references the link target is called "link source". There are two types of references: odx-links and short-name references.
The following descriptions are based on the implementation of the data model in XML.

7.3.13.2 REFERENCES VIA ODX-LINK

As a convention, ODX link elements are named by appending the suffix "–REF" to the name of the referenced element type, e.g. a UNIT-REF links to a UNIT. Link elements normally have no sub-elements and no text as content. The attributes DOCTYPE, DOCREF, ID-REF and REVISION are used to specify the link target.
ODX data may be assembled by composing so-called "document fragments". A document fragment represents a logical part of the ODX document. An ODX link usually references the link-target via the document fragment the link-target is located in. The document fragment is defined by the two attributes DOCTYPE and DOCREF.

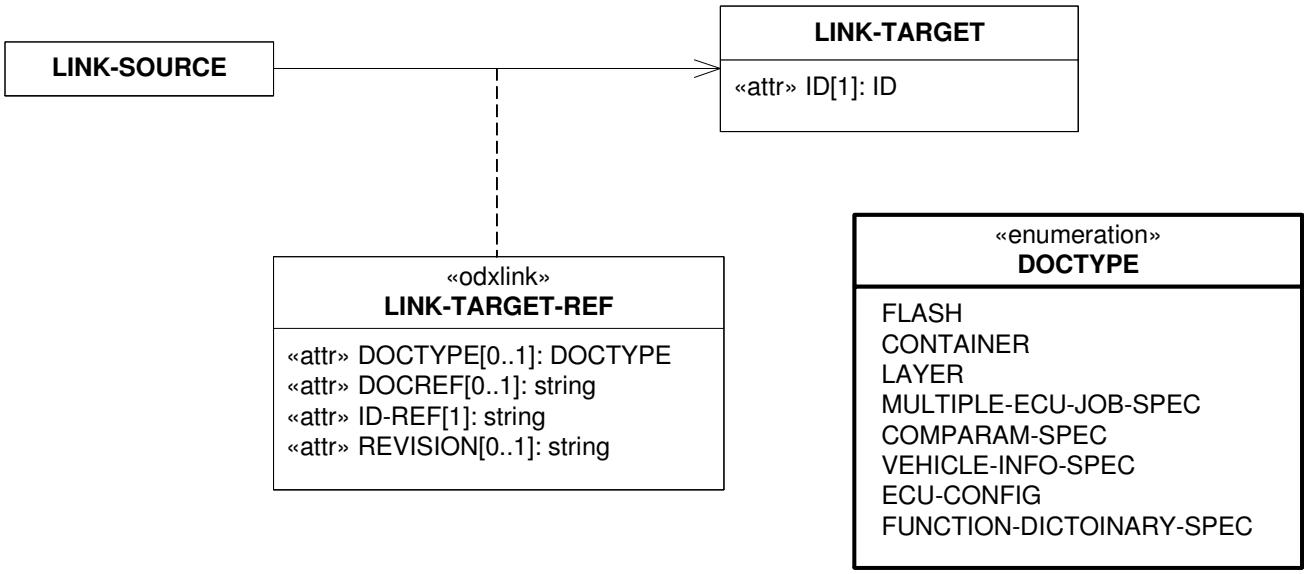


Figure 103 — Structure of an ODX link

Generally, ODX links are resolved in two steps:
gg) Find the document fragment that contains the link target

hh) Retrieve the link target from the document fragment

The first step uses the attributes DOCTYPE and DOCREF:
These two attributes may be defined only jointly. The DOCREF attribute refers to the SHORT-NAME of the ODX-CATEGORY or the DIAG-LAYER. If DOCREF and DOCTYPE are not set the link target exists in the local ODX-CATEGORY where also the link source exists (internal ODX link).

In the second step, the ID of the referenced element is specified by the ID-REF attribute. The optional REVISION attribute can be used to link to a specific version of the document fragment. An ODX conformant tool may ignore this attribute while resolving ODX links. The following examples show some use cases for ODX links.

EXAMPLE Linking a PROTOCOL layer

Container-based linking

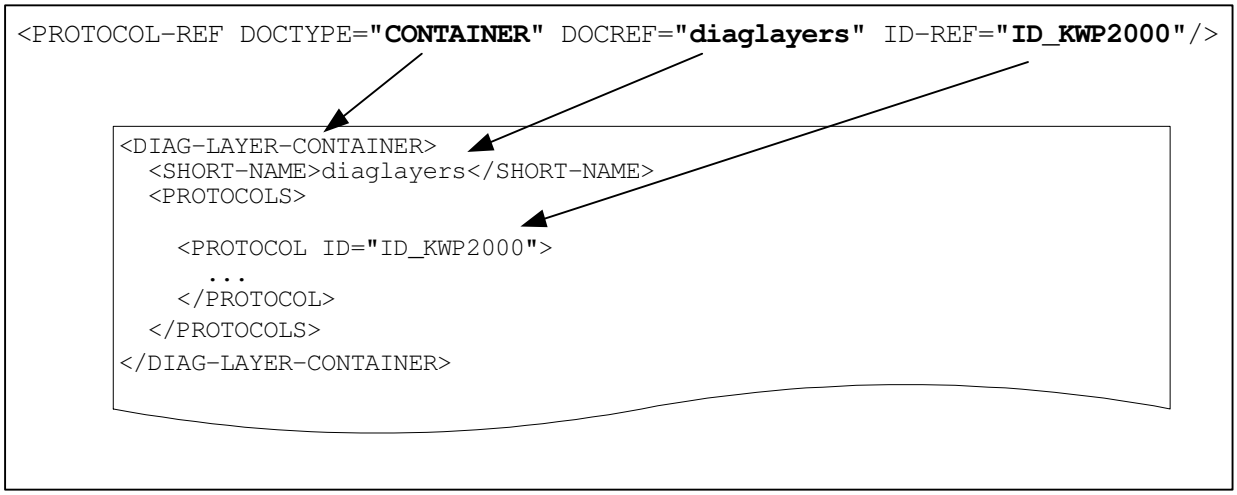


Figure 104 — Linking a PROTOCOL layer (via DIAG-LAYER-CONTAINER)

Layer-based linking

DOCREF refers to the SHORT-NAME of the document fragment while ID-REF refers to the ID.

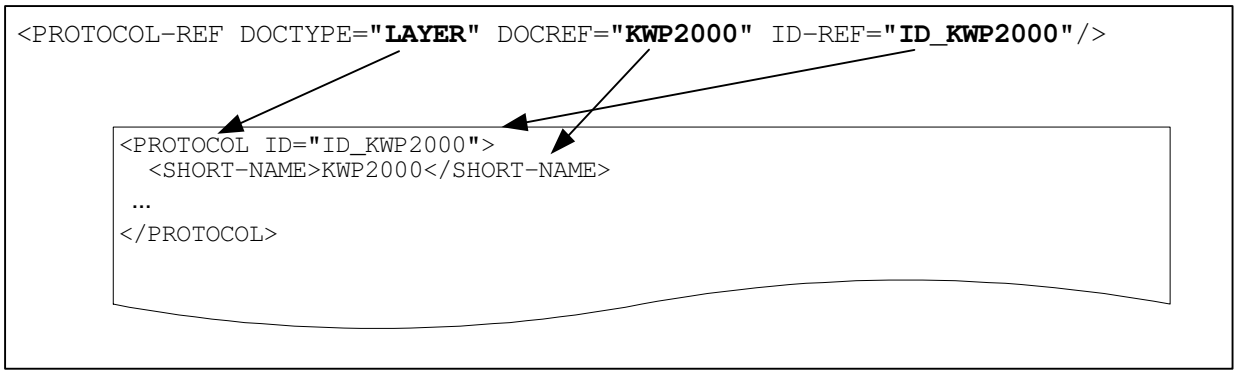


Figure 105 — Linking a PROTOCOL layer (via DIAG-LAYER)

EXAMPLE Linking a UNIT

Container-based linking

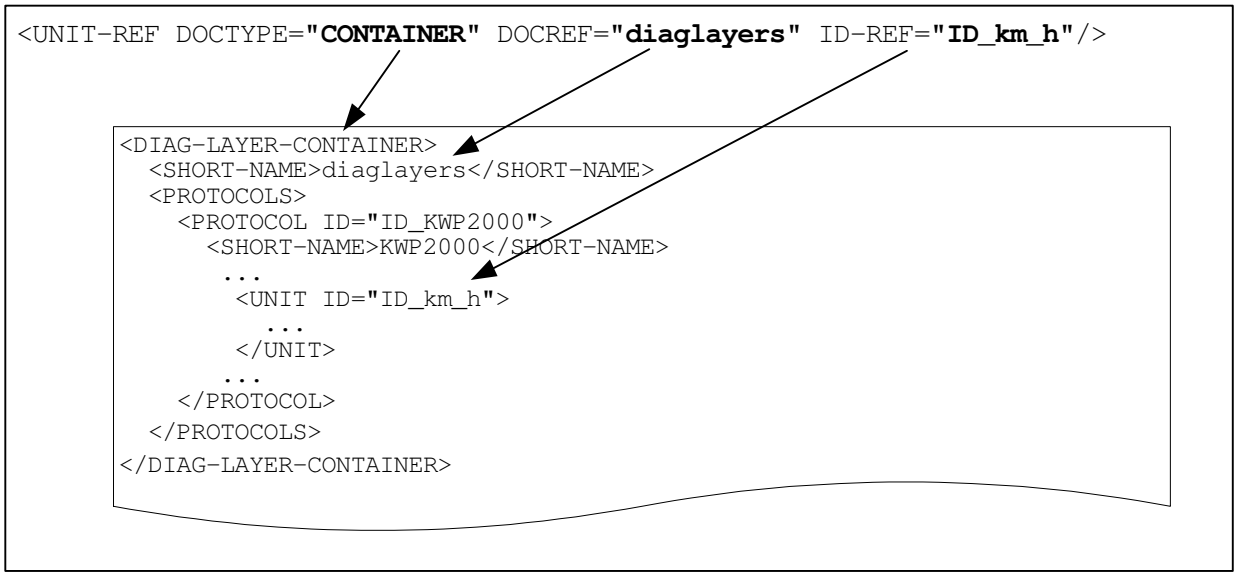


Figure 106 — Linking a UNIT (via DIAG-LAYER-CONTAINER)

Layer-based linking

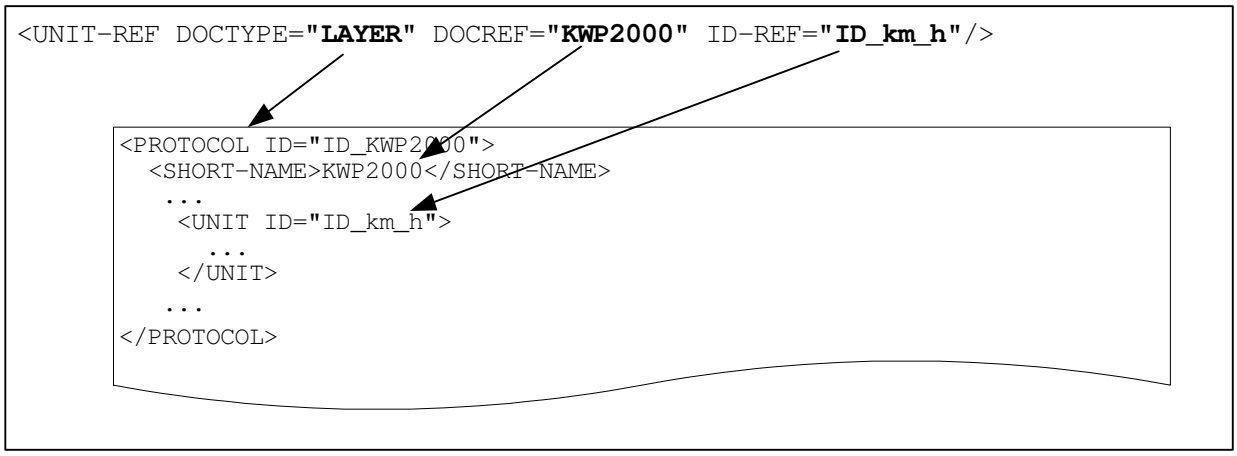


Figure 107 — Linking a UNIT (via DIAG-LAYER)

EXAMPLE Internal vs. external Linking

```
<DIAG-LAYER-CONTAINER>
  <SHORT-NAME>diaglayers</SHORT-NAME>
  <PROTOCOLS>
    <PROTOCOL ID="ISO14230">
      <SHORT-NAME>ISO14230</SHORT-NAME>
      ...
      <UNIT ID="ID_km_h">...</UNIT>
      ...
```

```
      <UNIT-REF DOCTYPE="CONTAINER" DOCREF="diaglayers" ID-REF="ID_km_h"/> (1)
      <UNIT-REF DOCTYPE="LAYER" DOCREF="ISO14230" ID-REF="ID_km_h"/>       (2)
      <UNIT-REF ID-REF="ID_km_h"/>                                         (3)
      ...
    </PROTOCOL>
  <PROTOCOLS>
  <ECU-SHARED-DATAS>
    <ECU-SHARED-DATA>
      ...
      <UNIT-REF DOCTYPE="CONTAINER" DOCREF="diaglayers" ID-REF="ID_km_h"/>
(4)
      <UNIT-REF ID-REF="ID_km_h"/>                                         (5)
            ...
    </ECU-SHARED-DATA>
  </ECU-SHARED-DATAS>
</DIAG-LAYER-CONTAINER>

<DIAG-LAYER-CONTAINER>
  <SHORT-NAME>someOtherContainer</SHORT-NAME>
  <ECU-SHARED-DATAS>
    <ECU-SHARED-DATA>
      ...
      <UNIT-REF DOCTYPE="LAYER" DOCREF="ISO14230" ID-REF="ID_km_h"/>       (6)
      <UNIT-REF DOCTYPE="CONTAINER" DOCREF="diaglayers" ID-REF="ID_km_h"/> (7)
      ...
    </ECU-SHARED-DATA>
  </ECU-SHARED-DATA>
</DIAG-LAYER-CONTAINER>
```

The links **(1)** to **(5)** are internal links.
The links **(6)** and **(7)** are external links.

7.3.13.3 REFERENCES VIA SHORT-NAME

Some elements of an ODX document like DIAG-SERVICE can be referenced via short-name. Parameters can for example reference a DOP via odx-link or short-name (stereotype in the UML drawings «odxlink-or-snref»). In case of short-name references the value inheritance and overwriting mechanism must be observed (see 7.3.2.4).

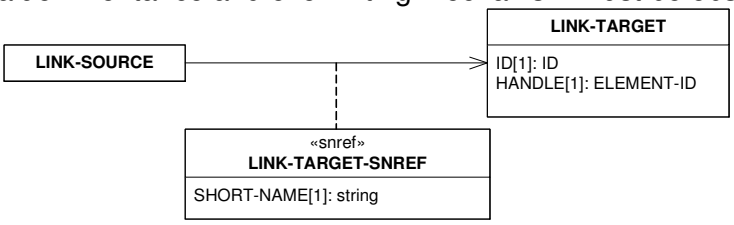


Figure 108 — Structure of a SNREF link

As a convention, short-name-references are named by appending the suffix "–SNREF" to the name of the referenced element type, e.g. PARAM-SNREF links to a parameter (PARAM).
The table bellow gives the summary of elements that are referenced via short-name:

Table 11 — Short-name referenced elements

Wrapper element	Referencing sub-element	Referenced element
MATCHING-PARAMETER	DIAG-COMM-SNREF	DIAG-SERVICE or SINGLE-ECU-JOB
MATCHING-PARAMETER	OUT-PARAM-IF-SNREF	PARAM of a response or OUTPUT-PARAM
MATCHING-COMPONENT	OUT-PARAM-IF-SNREF	PARAM of a response or OUTPUT-PARAM
ENV-DATA-DESC	PARAM-SNREF	PARAM of a response or OUTPUT-PARAM
NOT-INHERITED-DIAG-COMM	DIAG-COMM-SNREF	DIAG-SERVICE or SINGLE-ECU-JOB
NOT-INHERITED-VARIABLE	DIAG-VARIABLE-SNREF	DIAG-VARIABLE
COMM-RELATION	IN-PARAM-IF-SNREF	PARAM of a request or INPUT-PARAM
	OUT-PARAM-IF-SNREF	PARAM of a response or OUTPUT-PARAM
	DIAG-COMM-SNREF	DIAG-SERVICE, SINGLE-ECU-JOB or MULTIPLE-ECU-JOB
ACCESS-LEVEL	DIAG-COMM-SNREF	DIAG-SERVICE, SINGLE-ECU-JOB or MULTIPLE-ECU-JOB
SESSION-DESC	SESSION-SNREF	SESSION
	DIAG-COMM-SNREF	DIAG-SERVICE, SINGLE-ECU-JOB
IDENT-DESC	IDENT-IF-SNREF	IDENT
	DIAG-COMM-SNREF	DIAG-SERVICE, SINGLE-ECU-JOB
	OUT-PARAM-IF-SNREF	PARAM of a response or OUTPUT-PARAM
PARAM	DOP-SNREF	DOP-BASE
COMPARAM-REF	PROTOCOL-SNREF	PROTOCOL

EXAMPLE Referencing a response PARAM via short-name:

```
<OUT-PARAM-SNREF SHORT-NAME="ReadDataByAddress"/>
```

EXAMPLE Referencing a DIAG-SERVICE via short-name:

```
<DIAG-COMM-SNREF SHORT-NAME="DS_ReadDataByAddress"/>
```

7.3.13.4 BOUNDARY FOR UNIQUENESS OF SHORT-NAME

According to the reference mechanism via short-name and the value-inheritance mechanism adequate boundaries must be defined for unique short-names. Table 12 shows the boundary definitions of the short names at several positions. The term "global" means unique within one ODX database. It may be the scope of ODX data of a particular company as a whole. A group of elements in one table row means that all the short names of these elements together should be unique in the given boundary. The table defines the SHORT-NAME namespaces on the XML elements generated from the UML model and not for all UML elements. Therefore, e.g. child classes that inherit from a parent class with an <<impl-parent>> stereotype do not appear within this table.

EXAMPLE SESSION and DATABLOCK have to be unique within ECU-MEM but it is allowed that a SESSION has the same SHORT-NAME value as a DATABLOCK because they belong to different name spaces. Would they share the same name space a SESSION and a DATABLOCK with equal SHORT-NAME must not exist within one ECU-MEM.

Table 12 — Boundary for uniqueness of short-name

Name Space (Base Class)	Elements within Name Space	Boundary
BASIC-DOP	DATA-OBJECT-PROP, DTC-DOP, MUX, ENV-DATA, STRUCTURE, ENV-DATA-DESC, STATIC-FIELD, DYNAMIC-ENDMARKER-FIELD, DYNAMIC-LENGTH-FIELD, END-OF-PDU-FIELD	DIAG-DATADICTIONARY-SPEC
CASE	CASE, DEFAULT-CASE	MUX
CHECKSUM	CHECKSUM	SESSION
COMPANY-DATA	COMPANY-DATA	DIAG-LAYER
COMPARAM	COMPARAM	COMPARAM-SPEC
DATABLOCK	DATABLOCK	ECU-MEM
DIAG-COMM	DIAG-COMM, DIAG-SERVICE, SINGLE-ECU-JOB	DIAG-LAYER
DIAG-LAYER	PROTOCOL, FUNCTIONAL-GROUP, BASE-VARIANT, ECU-VARIANT, ECU-SHARED-DATA	Global
DIAG-VARIABLE	DIAG-VARIABLE	DIAG-LAYER
DTC	DTC	DTC-DOP
ECU-GROUP	ECU-GROUP	VEHICLE-INFORMATION
ECU-MEM-CONNECTOR	ECU-MEM-CONNECTOR	Global
ECU-MEM	ECU-MEM	Global
EXPECTED-IDENT	EXPECTED-IDENT	SESSION
FLASH-CLASS	FLASH-CLASS	ECU-MEM-CONNECTOR
FLASHDATA	FLASHDATA	ECU-MEM
FUNCT-CLASS	FUNCT-CLASS	DIAG-LAYER
INFO-COMPONENT	INFO-COMPONENT	VEHICLE-INFO-SPEC
LOGICAL-LINK	LOGICAL-LINK	VEHICLE-INFORMATION
MULTIPLE-ECU-JOB	MULTIPLE-ECU-JOB	Global
ODX-CATEGORY	COMPARAM-SPEC, MULTIPLE-ECU-JOB-SPEC, VEHICLE-INFO-SPEC, FLASH, DIAG-LAYER-CONTAINER, ECU-CONFIG, COMPARAM-SUBSET, FUNCTION-DICTIONARY	Global
OWN-IDENT	OWN-IDENT	DATABLOCK
PARAM	PARAM, INPUT-PARAM, OUTPUT-PARAM, NEG-OUTPUT-PARAM	REQUEST, RESPONSE, SINGLE-ECU-JOB, MULTIPLE-ECU-JOB
PHYSICAL-DIMENSION	PHYSICAL-DIMENSION	UNIT-SPEC
PHYSICAL-VEHICLE-LINK	PHYSICAL-VEHICLE-LINK	VEHICLE-INFORMATION
PHYS-MEM	PHYS-MEM	ECU-MEM
PHYS-SEGMENT	PHYS-SEGMENT	PHYS-MEM
PROJECT-IDENT	PROJECT-IDENT	ECU-MEM
REQUEST	REQUEST	DIAG-LAYER
RESPONSE	NEG-RESPONSE, POS-RESPONSE, GLOBAL-NEG-RESPONSE	DIAG-LAYER
SDG-CAPTION	SDG-CAPTION	SDGS
SEGMENT	SEGMENT	DATABLOCK
SESSION	SESSION	ECU-MEM
SESSION-DESC	SESSION-DESC	ECU-MEM-CONNECTOR
SW-VARIABLE	SW-VARIABLE	SW-VARIABLE
TABLE	TABLE	DIAG-DATADICTIONARY-SPEC
TABLE-ROW	TABLE-ROW	TABLE
TEAM-MEMBER	TEAM-MEMBER	COMPANY-DATA

Table 12 — Boundary for uniqueness of short-name

Name Space (Base Class)	Elements within Name Space	Boundary
UNIT-GROUP	UNIT-GROUP	UNIT-SPEC
UNIT	UNIT	UNIT-SPEC
VARIABLE-GROUP	VARIABLE-GROUP	DIAG-LAYER
VEHICLE-CONNECTOR	VEHICLE-CONNECTOR	VEHICLE-INFORMATION
VEHICLE-CONNECTOR-PIN	VEHICLE-CONNECTOR-PIN	VEHICLE-CONNECTOR
VEHICLE-INFORMATION	VEHICLE-INFORMATION	VEHICLE-INFO-SPEC
XDOC	XDOC	RELATED-DOC

7.4 USAGE SCENARIOS (DIAGNOSTIC)

7.4.1 DIAGNOSTIC SERVICE DESCRIPTION

As an example for this usage scenario the service InputOutputControlByLocalIdentifier from ISO 14230 is used. The request of this service has the following structure:

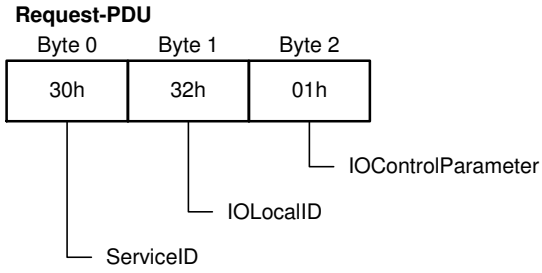


Figure 109 — Request of service 30h in ISO 14230

There are three parameters in the request:

- ServiceID (30h = InputOutputControlByLocalIdentifier),
- inputOutputLocalID (32h = "Desired Idle Adjustment")
- IOControlParameter (01h = reportCurrentState).

As a response the following PDU is possible:

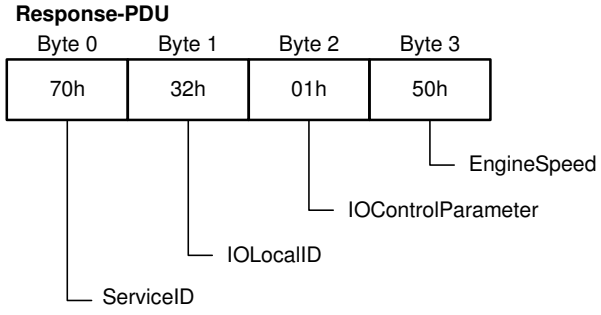


Figure 110 — Response of service 30h in ISO 14230

The first parameter of the response is the service id + 40h = 70h. The next two ones are the replication of the request parameters at the same positions. The last byte indicates the coded value of the current engine speed. The physical value is calculated by multiplying the coded one with 10. Thus the engine speed is equal 820 r/min (50h = 82; 82 * 10 = 820).
The diagnostic service is defined in ODX as follows:

```
<DIAG-SERVICE ID="DS_IOCBLID" SEMANTIC="CONTROL">
       <SHORT-NAME>DS_IOCBLID</SHORT-NAME>
       <AUDIENCE/>
       <REQUEST-REF ID-REF="REQ_IOCBLID"/>
       <POS-RESPONSE-REFS>
              <POS-RESPONSE-REF ID-REF="RESP_IOCBLID"/>
       </POS-RESPONSE-REFS>
</DIAG-SERVICE>
```

Whereby the request "REQ_IOCBLID" and the response "RESP_IOCBLID" look like this:

```
<REQUEST ID="REQ_IOCBLID">
       <SHORT-NAME>REQ_IOCBLID</SHORT-NAME>
       <PARAMS>
              <PARAM SEMANTIC="SERVICE-ID" xsi:type="CODED-CONST">
                     <SHORT-NAME>ServiceID</SHORT-NAME>
                     <BYTE-POSITION>0</BYTE-POSITION>
                     <CODED-VALUE>48</CODED-VALUE>
                     <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-
TYPE="A_UINT32">
                            <BIT-LENGTH>8</BIT-LENGTH>
                     </DIAG-CODED-TYPE>
              </PARAM>
              <PARAM xsi:type="CODED-CONST">
                     <SHORT-NAME>IOLocalID</SHORT-NAME>
                     <BYTE-POSITION>1</BYTE-POSITION>
                     <CODED-VALUE>50</CODED-VALUE>
                     <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-
TYPE="A_UINT32">
                            <BIT-LENGTH>8</BIT-LENGTH>
                     </DIAG-CODED-TYPE>
              </PARAM>
              <PARAM xsi:type="CODED-CONST">
                     <SHORT-NAME>IOControlParameter</SHORT-NAME>
                     <BYTE-POSITION>2</BYTE-POSITION>
                     <CODED-VALUE>01</CODED-VALUE>
                     <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-
TYPE="A_UINT32">
                            <BIT-LENGTH>8</BIT-LENGTH>
                     </DIAG-CODED-TYPE>
              </PARAM>
       </PARAMS>
</REQUEST>

<POS-RESPONSE ID="RESP_IOCBLID">
       <SHORT-NAME>RESP_IOCBLID</SHORT-NAME>
       <PARAMS>
              <PARAM xsi:type="CODED-CONST" SEMANTIC="SERVICE-ID">
                     <SHORT-NAME>ServiceID</SHORT-NAME>
                     <BYTE-POSITION>0</BYTE-POSITION>
                     <CODED-VALUE>112</CODED-VALUE>
                     <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-
TYPE="A_UINT32">
                            <BIT-LENGTH>8</BIT-LENGTH>
                     </DIAG-CODED-TYPE>
              </PARAM>
              <PARAM xsi:type="MATCHING-REQUEST-PARAM">
                     <SHORT-NAME>IOLocalID</SHORT-NAME>
                     <BYTE-POSITION>1</BYTE-POSITION>
                     <REQUEST-BYTE-POS>1</REQUEST-BYTE-POS>
```

```
                <BYTE-LENGTH>1</BYTE-LENGTH>
            </PARAM>
            <PARAM xsi:type="MATCHING-REQUEST-PARAM">
                <SHORT-NAME>IOControlParameter</SHORT-NAME>
                <BYTE-POSITION>2</BYTE-POSITION>
                <REQUEST-BYTE-POS>2</REQUEST-BYTE-POS>
                <BYTE-LENGTH>1</BYTE-LENGTH>
            </PARAM>
            <PARAM xsi:type="VALUE">
                <SHORT-NAME>EngineSpeed</SHORT-NAME>
                <BYTE-POSITION>3</BYTE-POSITION>
                <DOP-REF ID-REF="DOP_EngineSpeed"/>
            </PARAM>
        </PARAMS>
</POS-RESPONSE>
```

In the response the second and the third parameters are declared as "MATCHING-REQUEST-PARAM" parameters. Therefore, they refer to the appropriate byte position in the request by the element REQUEST-BYTE-POS. The firth parameter is the proper result of the request. It contains a link to a DOP, which is responsible for conversion. This DOP and the associated UNIT are defined as follows:

```
<DATA-OBJECT-PROP ID="DOP_EngineSpeed">
        <SHORT-NAME>DOP_EngineSpeed</SHORT-NAME>
        <COMPU-METHOD>
            <CATEGORY>LINEAR</CATEGORY>
            <COMPU-INTERNAL-TO-PHYS>
                <COMPU-SCALES>
                    <COMPU-SCALE>
                        <COMPU-RATIONAL-COEFFS>
                            <COMPU-NUMERATOR>
                                <V>0</V>
                                <V>10</V>
                            </COMPU-NUMERATOR>
                            <COMPU-DENOMINATOR>
                                <V>1</V>
                            </COMPU-DENOMINATOR>
                        </COMPU-RATIONAL-COEFFS>
                    </COMPU-SCALE>
                </COMPU-SCALES>
            </COMPU-INTERNAL-TO-PHYS>
        </COMPU-METHOD>
        <DIAG-CODED-TYPE BASE-DATA-TYPE="A_INT32" xsi:type="STANDARD-LENGTH-TYPE">
            <BIT-LENGTH>8</BIT-LENGTH>
        </DIAG-CODED-TYPE>
        <PHYSICAL-TYPE BASE-DATA-TYPE="A_INT32" DISPLAY-RADIX="DEC"/>
        <UNIT-REF ID-REF="Unit_RPerMin"/>
</DATA-OBJECT-PROP>


<UNIT ID="Unit_RPerMin">
        <SHORT-NAME>Unit_RPerMin</SHORT-NAME>
        <DISPLAY-NAME>r/min</DISPLAY-NAME>
        <PHYSICAL-DIMENSION-REF ID-REF="PD_RPerMin"/>
</UNIT>


<PHYSICAL-DIMENSION ID="PD_RPerMin">
        <SHORT-NAME>PD_RPerMin</SHORT-NAME>
        <TIME-EXP>-1</TIME-EXP>
</PHYSICAL-DIMENSION>
```

The DOP's COMPU-METHOD is of type "LINEAR" which is indicated by the element CATEGORY. The numerator is defined by two V elements (<V>0</V><V>10</V>). This leads to the formula:

PhysicalValue = 0 + CodedValue * 10
The referenced UNIT object provides the units of the computed value (r/min). It is derived from the physical dimension "PHYS-DIM-rPerMin" with the time exponent value of -1.

7.4.2 DYNAMICALLY DEFINED MESSAGES

With some protocols (e.g. ISO 14230 or ISO 14229-1) it is possible to dynamically define data records at run time, which contain values from other records identified by a local identifier (LID), a common identifier (CID) or an address in the ECU memory, etc. These new records are identified by the dynamic identifier (DYN-ID). In ISO 14230 specification this identifier is called "dynamically defined local identifier" (DDLID).
The figure below shows the process of creating a new data record with ISO 14230 by referencing values from other records identified by LIDs. The record with LID=01h consists of three values whereby the other one (LID=02h) contains two values. One value from the first record and one value from second one are grouped into the new record with DYN-ID=F0h.

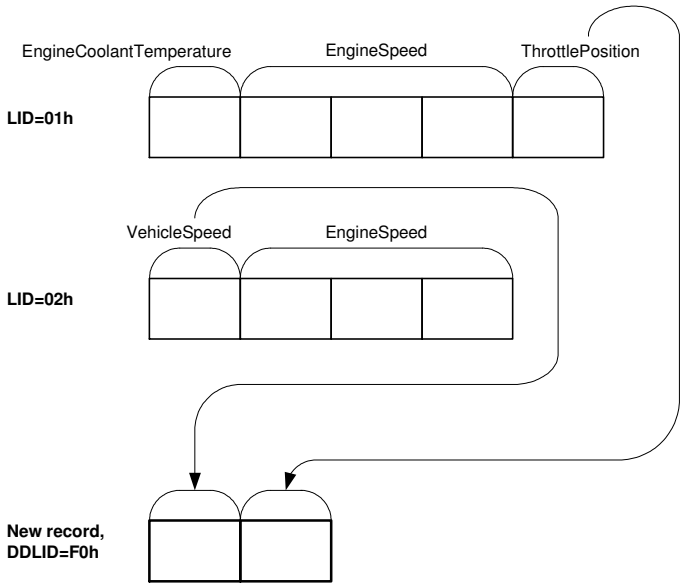


Figure 111 — Creating of a new data record in ISO 14230

The next figure shows an example of a DYN-DEFINED-SPEC object. For creating a new data record in the definition mode "LOCAL-ID" the table TAB_LIDs and one of the DYN-IDs F0h or F1h are to be used. To combine COMMON-IDs to one DYN-ID the table TAB_CIDs and one of the DYN-IDs F2h or F3h are used. The table with SEMANTIC="LOCAL-ID" holds the description of all records identified by a local identifier and the table with SEMANTIC="COMMON-ID" defines data records identified by a common identifier. In the first table, TABLE-ROW with KEY=01h refers to the structure LID_01, i.e. the data record with LID=01h is given by the structure LID_01. The latter consists of three values (parameters): EngineCoolantTemperature, EngineSpeed and ThrottlePosition. Other table rows are interpreted in the same way.

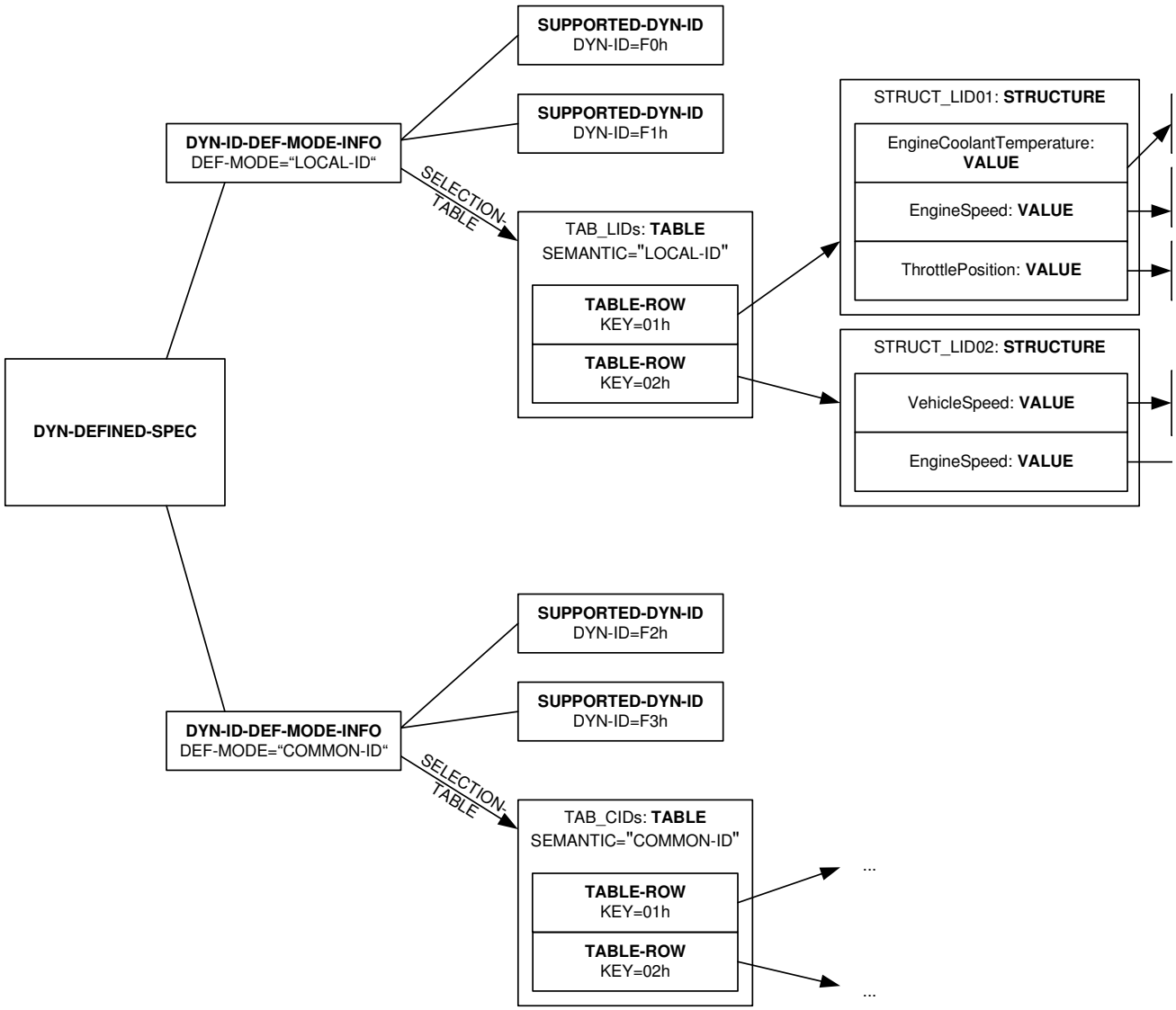


Figure 112 — Example of DYN-DEFINED-SPEC

The member DIAGNOSTIC-CLASS of DIAG-COMM is used to indicate the runtime system that the service/job defines a new DYN-ID (DIAGNOSTIC-CLASS="DYN-DEF-MESSAGE"), that the service/job reads data using the DYN-ID defined before (DIAGNOSTIC-CLASS="READ-DYN-DEF-MESSAGE") or that the service/job clears information about the DYN-ID definitions (DIAGNOSTIC-CLASS="CLEAR-DYN-DEF-MESSAGE"). In the following examples the ISO 14230 service with ID=21h has the DIAGNOSTIC-CLASS "READ-DYN-DEF-MESSAGE". There are also two services with ID=2Ch. One of them has the DIAGNOSTIC-CLASS "DYN-DEF-MESSAGE" and the other one "CLEAR-DYN-DEF-MESSAGE". A complete implementation of these services can be found in Annex C.2.

Dynamic case

In the first example, the user constructs the DDLID (assisted by the application) at run time.

Figure 113 shows an example for the process of defining of DYN-IDs with the service 0x2C. The request of the service contains the parameters ServiceID (=2Ch) and DDLID (it is filled with the new DYN-ID by the user at run time). The last parameter references an

END-OF-PDU-FIELD, which references the structure that is used to define an item for the new record. This structure is repeated for each new item in the defined DYN-ID record. The user fills the values in this structure according to the used protocol (in this example, it is ISO 14230).
If a DYN-DEF-MESSAGE is to be sent to the ECU, the application can offer the user to choose the following items:

- One of the definition modes
- One of the supported DYN-IDs in the selected definition mode,
- A specific row of the TABLE that belongs to the selected definition mode,
- Concrete values of the structures referenced by the selected row (in this example these are ThrottlePosition and VehicleSpeed).

This information is used to fill the structure referenced from the END-OF-PDU-FIELD. Alternatively, an advanced user can fill this structure manually, e.g. if the application does not provide an assistant for assembling of DYN-ID records. In Figure 113, this structure is called "STRUCT_DDLIDItem" and it is filled two times meaning that two values are combined to one record with DYN-ID=F0h.

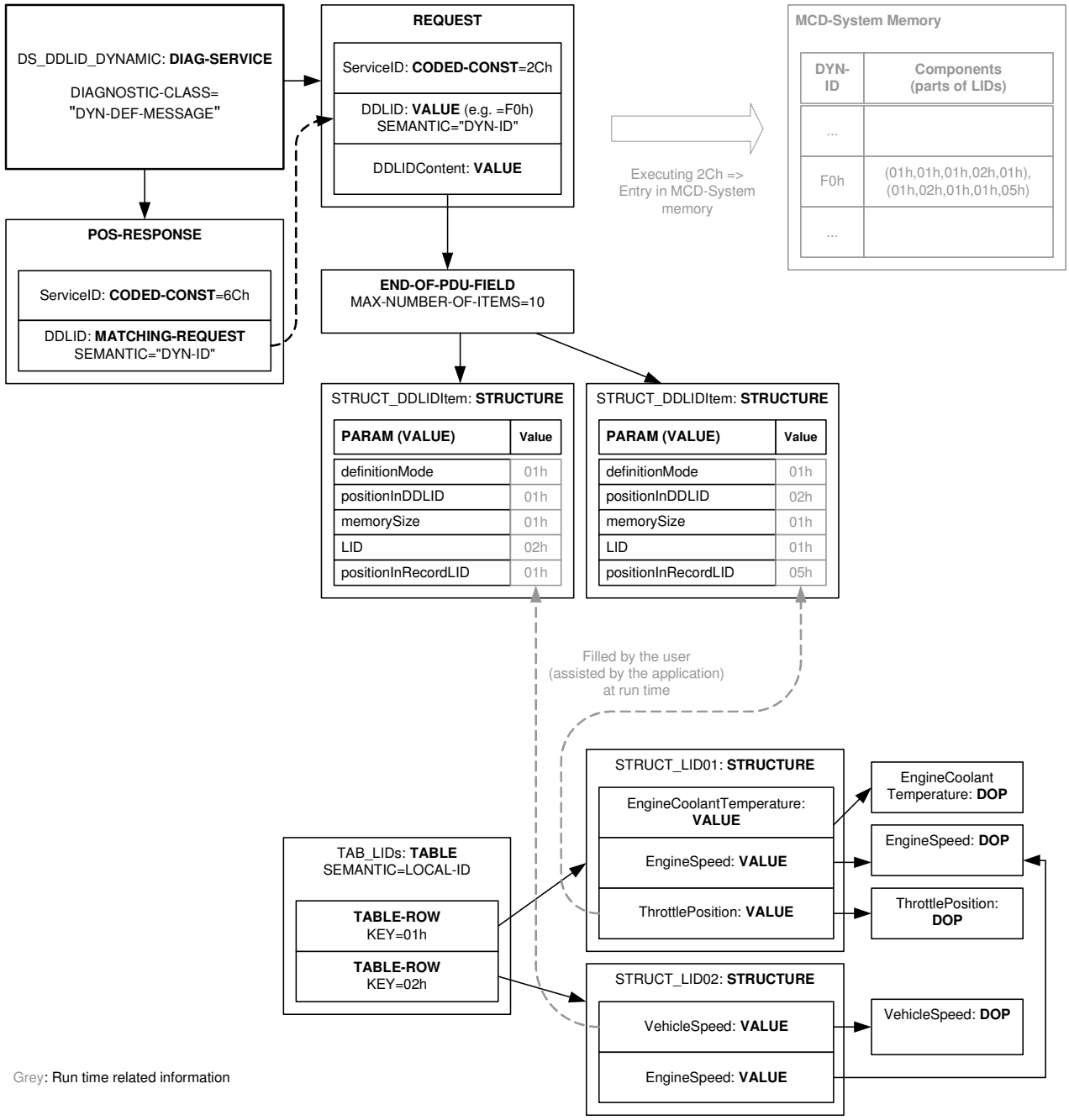


Figure 113 — Example of DYN-DEF-MESSAGE, dynamic case

After executing the DYN-DEF-MESSAGE service the MCD-System has to store the following information:

— DYN-ID (in ISO 14230: DDLID)

— Information about components of the DYN-ID record depending on the used protocol (in ISO 14230: definition mode, position in LID, memory size, LID and position in DDLID)

This information is sufficient to interpret the DYN-ID record at a later time.

Now the defined DYN-ID record can be read by the READ-DYN-DEF-MESSAGE service (21h in ISO 14230) like shown in the following figure:

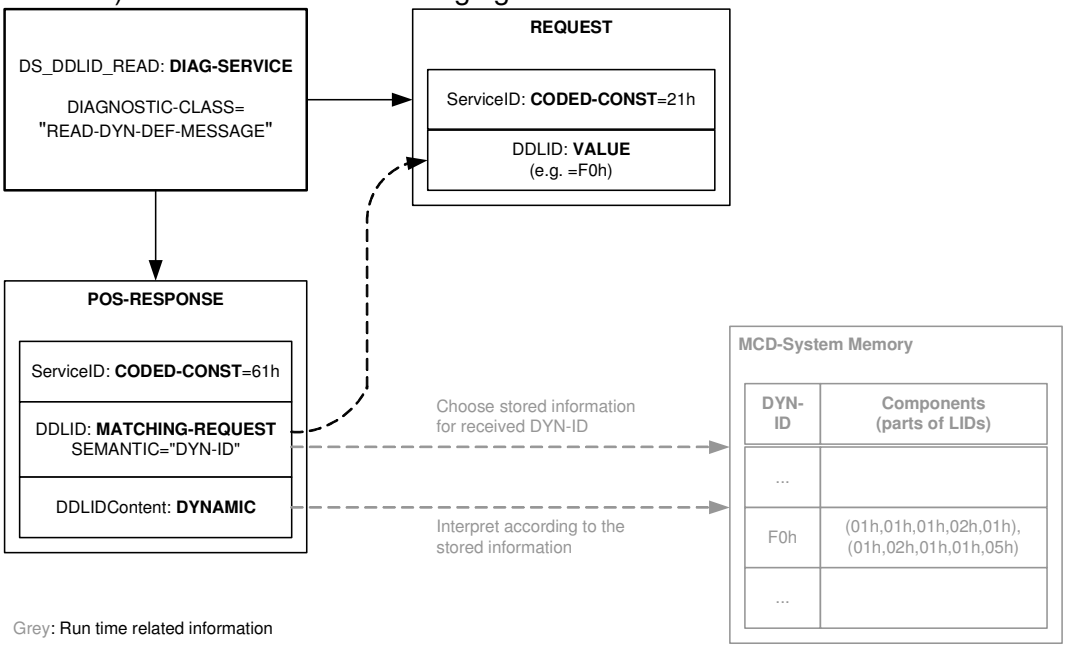


Figure 114 — Example of READ-DYN-DEF-MESSAGE, static and dynamic cases

In the response a parameter with SEMANTIC="DYN-ID" indicates that the entry with DYN-ID=F0h, stored in the memory, should be chosen for the following interpretation of data. The next parameter (of type DYNAMIC) has no reference to any DOP, because its interpretation is only known at run time and the data stored in the memory is to be used for it.
The MCD-system does not have any information about the TABLE used for definition of the DYN-ID record (this information has only the application). The MCD-System has only the information about the content of the DYN-ID record. In this example, the following information is available for each component of the DYN-ID record:

— SEMANTIC of the TABLE (derived from the parameter "defninitionMode" in the structure STRUCT_DDLIDItem)

— Identifier (parameter LID in the structure STRUCT_DDLIDItem)

— The remaining parameters used for the definition of the DYN-ID record (in this example, the remaining parameters of the structure STRUCT_DDLIDItem)

All TABLEs listed in DYN-DEFINED-SPEC with the given SEMANTIC are searched for the row with the key that is equal to the given LID. The first row found is then used to interpret the received data in the way it was defined by the DYN-DEF-MESSAGE service. In this way the read service can be coded in ODX without knowing which LID's (or CID's etc.) the user will group during run time. The dynamic part of the service must be implemented by the runtime system and is dependant on the diagnosis protocol.
Static case

The author of an ODX instance can also assign a set of data to a DYN-ID defining a DYN-DEF-MESSAGE service with fixed values in the structure that represent a data item. If *n* is the number of items grouped to a DYN-ID record, *n* structures are referenced from *n* parameters of the request (in Figure 115 *n*=2). Each structure has parameters of type CODED-CONST. The parameter count can be different in different protocols. In ISO 14230 there are five ones: definitionMode, positionInDDLID, memorySize, LID and positionInRecordLID.

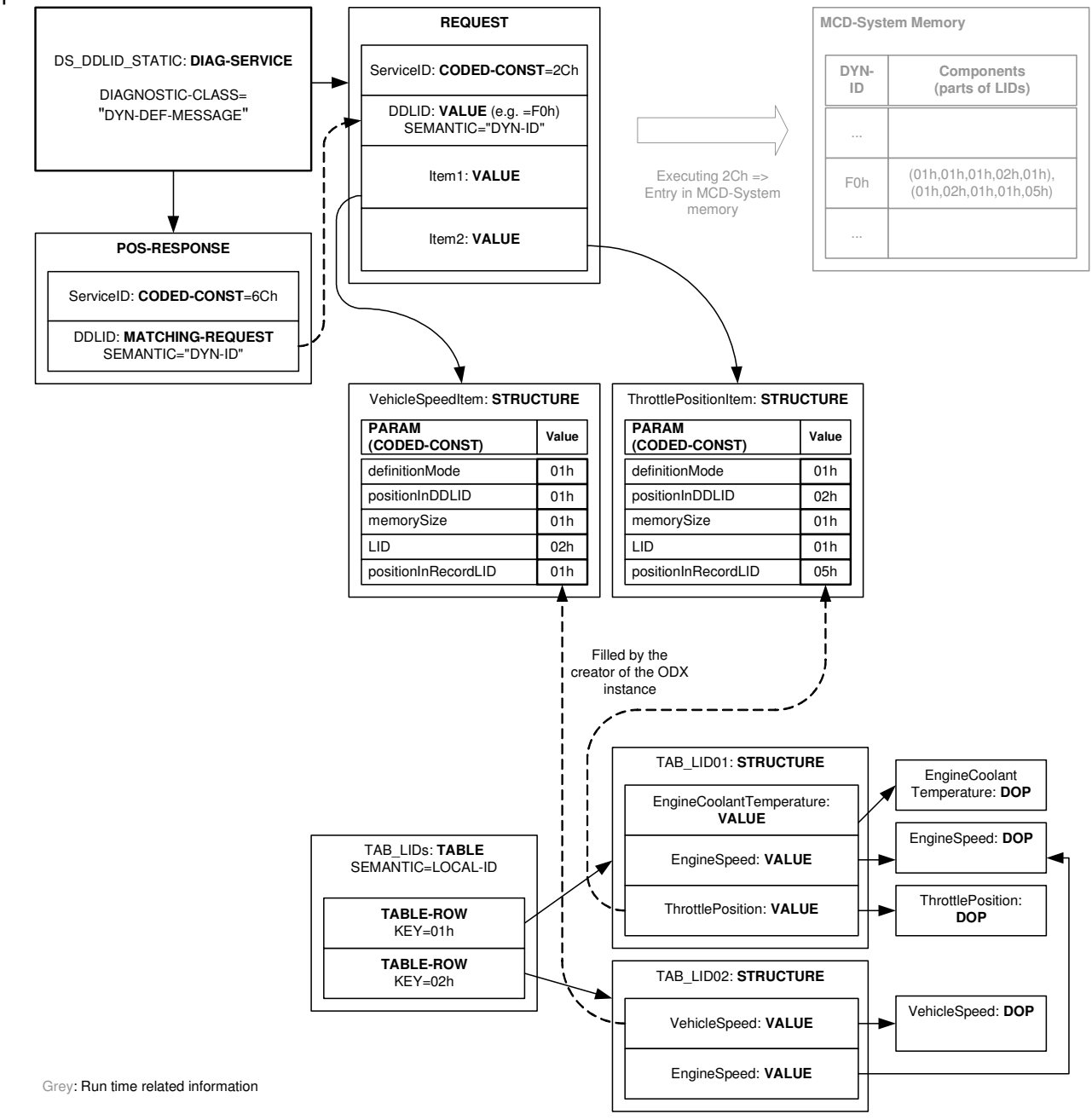


Figure 115 — Example of DYN-DEF-MESSAGE, static case

The execution of the service occurs in the same manner as in the dynamic case. Information about DYN-ID and its content is stored in the memory again. The READ-DYN-DEF-MESSAGE service can be used without any changes to read this DYN-ID record. From the point of view of the runtime system there is no difference between the dynamic and the static case.

7.4.3 VARIANT IDENTIFICATION

Within ODX, different variants of an ECU can be managed. Only one ECU variant (either ECU base variant or ECU variant) may be active at one time during runtime. This is necessary, since they are all identified via the same physical address. Before a variant identification session is started, the user has to select a BASE-VARIANT. Afterwards, the identification of an ECU-VARIANT performs by execution of DIAG-COMMs. Whether it is necessary to send one or more DIAG-COMM depends on the Protocol and the ECU implementation. The identification itself is performed by comparison of one or more results returned by the ECU with the variant identification data in the ECU-VARIANT instances.

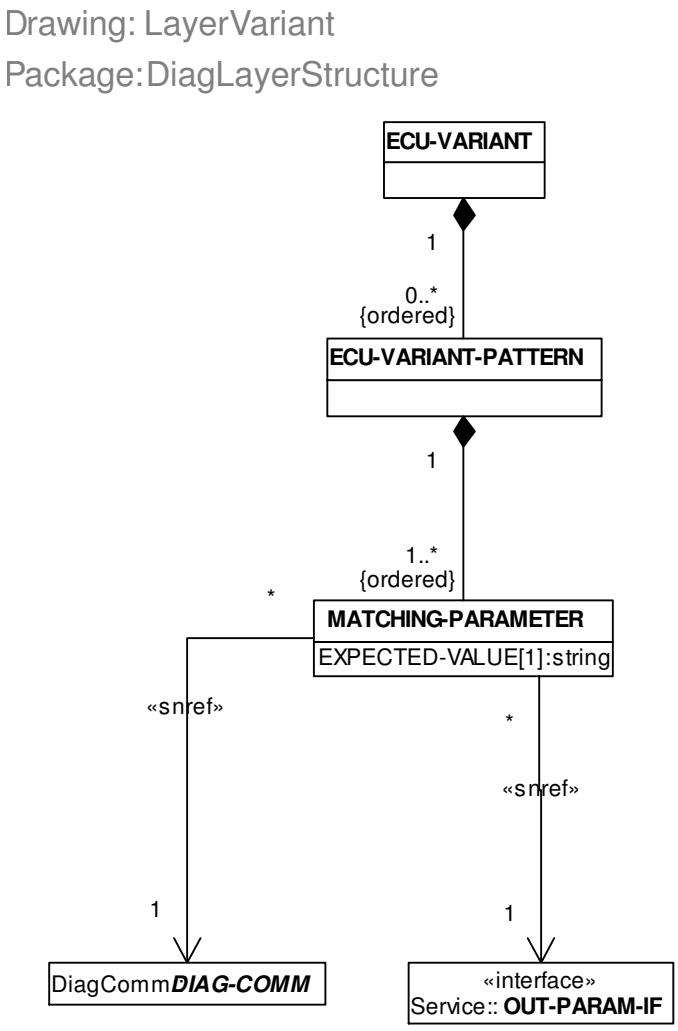


Figure 116 — Structure of variant identification

ECU-VARIANT-PATTERNs are organised in a list corresponding to the different ECU-VARIANTs that can be described via the ODX document. More than one ECU-VARIANT-PATTERN might be necessary if different ECU samples (with different identifications) do not differ regarding their diagnostic behaviour and are defined as one ECU-VARIANT in the ODX container. Every ECU-VARIANT-PATTERN contains a list of MATCHING-PARAMETER with values identifying the current ECU. All ECU-VARIANT-PATTERN are disjoint, each of them identifies this ECU-VARIANT. The different MATCHING-PARAMETER might be determined via more than one DIAG-COMM. The ECU-VARIANT-PATTERNs are only valid for one ECU-Variant layer. In case the DIAG-COMM is a DIAG-SERVICE and it contains more than one response (positive or negative), the parameter referenced by MATCHING-PARAMETER through OUT-PARAM-IF also makes in implicit statement as to which response is expected. If another response than the expected one is returned at runtime, the variant is not successfully matched. At least one POS-RESPONSE must contain the referenced parameter. If the parameter is contained in a POS-RESPONSE, it must have a unique SHORT-NAME. Otherwise the data definition might lead to an undefined behaviour of the system. This parameter must additionally refer to a DATA-OBJECT-PROP.
The variant identification algorithm works in the following way: All variants belonging to the same base variant are stored in an ordered list depending on the rising alphabetical order of their SHORT-NAMEs (this ensures deterministic behavior between different implementations of a runtime system). The first ECU-VARIANT's patterns are checked first, again in the order of their appearance within the ECU-VARIANT.
A MATCHING-PARAMETER matches if the EXPECTED-VALUE and the physical value of the response parameter referenced with the OUT-PARAM-IF-SNREF are equal. An ECU-VARIANT-PATTERN matches, if all it's MATCHING-PARAMETERs match. A variant is detected if at least one ECU-VARIANT-PATTERN matches. As soon as one ECU-VARIANT matched, the variant identification terminates. The variant is selected. If no variant could be identified, an exception occurs.

EXAMPLE Defining a set of ECU-VARIANT-PATTERN

DIAG-COMM	DS_IdentificationRead	
	Request: 0x1A, 0x86	
	Response: 0x5A, 0x86, 0x16, 0x80, 0x03	
Pattern 1	ExpectedValue = Supplier1	
	ExpectedValue = 32769	
Pattern 2	ExpectedValue = Supplier2	(CodedValue = 0x16)
	ExpectedValue = 32771	(Coded-Value = 0x8003)

ECU Variant2 using Pattern 2:

```
<ECU-VARIANT ID="Variant2">
       <SHORT-NAME>Variant2</SHORT-NAME>
       <ECU-VARIANT-PATTERNS>
              <ECU-VARIANT-PATTERN>
                     <MATCHING-PARAMETERS>
                            <MATCHING-PARAMETER>
                                   <EXPECTED-VALUE>Supplier2</EXPECTED-VALUE>
                                   <DIAG-COMM-SNREF SHORT-
NAME="DS_IdentificationRead"/>
                                   <OUT-PARAM-IF-SNREF SHORT-
NAME="SupplierIdentification"/>
                            </MATCHING-PARAMETER>
```

```
                        <MATCHING-PARAMETER>
                              <EXPECTED-VALUE>32771</EXPECTED-VALUE>
                              <DIAG-COMM-SNREF SHORT-
NAME="DS_IdentificationRead"/>
                              <OUT-PARAM-IF-SNREF SHORT-
NAME="DiagnosticVersion"/>
                        </MATCHING-PARAMETER>
                  </MATCHING-PARAMETERS>
            </ECU-VARIANT-PATTERN>
      </ECU-VARIANT-PATTERNS>
      <PARENT-REFS>
            <PARENT-REF ID-REF="ECU_xyz" xsi:type="BASE-VARIANT-REF"/>
      </PARENT-REFS>
</ECU-VARIANT>
```

Positive Response of DIAG-COMM "IdentificationRead":

```
<POS-RESPONSE ID="RESP_IdentificationRead">
      <SHORT-NAME>RESP_IdentificationRead</SHORT-NAME>
      <PARAMS>
            <PARAM SEMANTIC="DATA" xsi:type="VALUE">
                  <SHORT-NAME>SupplierIdentification</SHORT-NAME>
                  <BYTE-POSITION>2</BYTE-POSITION>
                  <DOP-REF ID-REF="DOP_SupplierList"/>
            </PARAM>
            <PARAM SEMANTIC="DATA" xsi:type="VALUE">
                  <SHORT-NAME>DiagnosticVersion</SHORT-NAME>
                  <BYTE-POSITION>3</BYTE-POSITION>
                  <DOP-REF ID-REF="DOP_2ByteHexDump"/>
            </PARAM>
      </PARAMS>
</POS-RESPONSE>
```

DOP SupplierIdentification:

```
<DATA-OBJECT-PROP ID="DOP_SupplierList">
      <SHORT-NAME>DOP_SupplierList</SHORT-NAME>
      <COMPU-METHOD>
            <CATEGORY>TEXTTABLE</CATEGORY>
            <COMPU-INTERNAL-TO-PHYS>
                  <COMPU-SCALES>
                        <COMPU-SCALE>
                              <LOWER-LIMIT INTERVAL-TYPE="CLOSED">21</LOWER-
LIMIT>
                              <UPPER-LIMIT INTERVAL-TYPE="CLOSED">21</UPPER-
LIMIT>
                              <COMPU-CONST>
                                    <VT>Supplier1</VT>
                              </COMPU-CONST>
                        </COMPU-SCALE>
                        <COMPU-SCALE>
                              <LOWER-LIMIT INTERVAL-TYPE="CLOSED">22</LOWER-
LIMIT>
                              <UPPER-LIMIT INTERVAL-TYPE="CLOSED">22</UPPER-
LIMIT>
                              <COMPU-CONST>
                                    <VT>Supplier2</VT>
                              </COMPU-CONST>
                        </COMPU-SCALE>
                  </COMPU-SCALES>
            </COMPU-INTERNAL-TO-PHYS>
      </COMPU-METHOD>
      <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-
TYPE">
```

```
            <BIT-LENGTH>8</BIT-LENGTH>
        </DIAG-CODED-TYPE>
        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
</DATA-OBJECT-PROP>
```

DOP DiagnosticVersion:

```
<DATA-OBJECT-PROP ID="DOP_2ByteHexDump">
        <SHORT-NAME>DOP_2ByteHexDump</SHORT-NAME>
        <COMPU-METHOD>
            <CATEGORY>IDENTICAL</CATEGORY>
        </COMPU-METHOD>
        <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-
TYPE">
            <BIT-LENGTH>16</BIT-LENGTH>
        </DIAG-CODED-TYPE>
        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32" DISPLAY-RADIX="HEX"/>
</DATA-OBJECT-PROP>
```

7.4.4 BASE VARIANT IDENTIFICATION SCENARIO

This chapter summarizes how data for a base variant identification scenario can be authored and what the runtime semantics of these data structures are. Base variant identification is applicable to all situations where an alternative built-in of ECUs needs to be resolved. Alternative ECUs are such that exactly or at most one of a set of possible base variants is actually built into a vehicle. Typical examples are different engine controllers for different engines or different radio ECUs from the low-end to the high-end.

7.4.4.1 ODX DATA STRUCTURES FOR BASE-VARIANT IDENTIFICATION SCENARIO

To group alternative BASE-VARIANTs together, the ECU-GROUP element is used. Through the GROUP-MEMBER class it references BASE-VARIANTs. For every reference it can also be specified which LOGICAL-LINK is to be used when resolving this variant. Resolution via functional addressing and resolution via physical addressing are both supported. By consequence, two possible references to LOGICAL-LINKs are supported: A physical resolution link and a functional resolution link. The following constraints apply:

1. One of the two link types must be defined for every GROUP-MEMBER. The specification of both kinds of links is equally possible
2. The FUNCT-RESOLUTION-LINK points to a LOGICAL-LINK that references a FUNCTIONAL-GROUP and no BASE-VARIANT.
3. The PHYS-RESOLUTION-LINK points to a LOGICAL-LINK that references a BASE-VARIANT.

To define how a particular BASE-VARIANT within the group can be identified by communication with the vehicle, every BASE-VARIANT can contain a BASE-VARIANT-PATTERN. A BASE-VARIANT-PATTERN contains a set of MATCHING-PARAMETERs, which contains an EXPECTED-VALUE. This EXPECTED-VALUE is matched against the content of a referenced DIAG-COMM's parameter referenced through OUT-PARAM-IF. If the values are equal, the MATCHING-PARAMETER is considered as matched. If all MATCHING-PARAMETERs of one BASE-VARIANT-PATTERN are considered as matched, the whole pattern is considered as matched and the BASE-VARIANT is identified.
Every MATCHING-PARAMETER also has an attribute that is to be used to determine whether the DIAG-COMM to retrieve the actual value from the vehicle is to be executed with physical or functional addressing (USE-PHYSICAL-ADDRESSING), with the default being physical addressing. If the referenced DIAG-COMM only supports one addressing mode, the value of this attribute is to be ignored. The following constraints apply:

1. If the BASE-VARIANT-PATTERN of BASE-VARIANT references at least one MATCHING-PARAMETER with USE-PHYSICAL-ADDRESSING = false, the BASE-VARIANT must have a FUNCTIONAL-RESOLUTION-LINK specified in every ECU-GROUP it appears in.
2. If the BASE-VARIANT-PATTERN of BASE-VARIANT references at least one MATCHING-PARAMETER with USE-PHYSICAL-ADDRESSING = true, the BASE-VARIANT must have a PHYS-RESOLUTION-LINK specified in every ECU-GROUP it appears in.

Together with the referenced OUT-PARAM an implicit statement is made as to which kind of response is expected (also refer to variant identification, where this mechanism also applies). For example, if the OUT-PARAM-IF references the negative response code of a negative response and at runtime a positive response is received, the MATCHING-PARAMETER is considered to be not matched.

7.4.4.2 Sequence of Events for BASE-VARIANT identification

To explain the semantics of base variant identification in a more detailed manner, the defined data is to be interpreted as follows:

1. The BASE-VARIANTs are iterated in their order within the ECU-GROUP
2. For the current BASE-VARIANT: Iterate through the MATCHING-PARAMETERs in the order they are defined in the BASE-VARIANT-PATTERN
   a. As soon as one MATCHING-PARAMETER is not matched, the BASE-VARIANT is considered as not identified; Continue with 1.
   b. As soon as all MATCHING-PARAMETERs within the BASE-VARIANT-PATTERN of the BASE-VARIANT under investigation matched, that BASE-VARIANT is considered identified – Continue with 3.
3. If a BASE-VARIANT was identified successfully, the ACCESS-KEY of that BASE-VARIANT is returned; in all other cases an exception occurs.

7.4.4.3 ODX Example

The following example shows a DIAG-LAYER-CONTAINER with one FUNCTIONAL-GROUP and two BASE-VARIANTs inheriting from it. Both BASE-VARIANTs contain a BASE-VARIANT-PATTERN which matches the Software Version Number against an expected value. The DIAG-SERVICE to read the identification value is authored on the level of the FUNCTIONAL-GROUP. It is inherited as-is by both BASE-VARIANTs. As it has an ADDRESSING mode of FUNCTIONAL-OR-PHYSICAL it can be used for both functional and physical base variant identification. In the given example BV1 shall be resolved functionally, while BV2 shall be resolved physically (cf. value of USE-PHYSICAL-ADDRESSING element).

```
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=" odx.xsd" MODEL-VERSION="2.1.0">
       <DIAG-LAYER-CONTAINER ID="DLC_10423">
              <SHORT-NAME>DLC_A</SHORT-NAME>
              <FUNCTIONAL-GROUPS>
                     <FUNCTIONAL-GROUP ID="FG_10424">
                            <SHORT-NAME>FG</SHORT-NAME>
                            <DIAG-DATA-DICTIONARY-SPEC>
                                   <DATA-OBJECT-PROPS>
                                          <DATA-OBJECT-PROP ID="DOP_11081">
                                                 <SHORT-NAME>HexByteField</SHORT-
NAME>
                                                 <COMPU-METHOD>

       <CATEGORY>IDENTICAL</CATEGORY>
                                                 </COMPU-METHOD>
                                                 <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_BYTEFIELD" TERMINATION="END-OF-PDU" xsi:type="MIN-MAX-LENGTH-TYPE">
                                                        <MAX-LENGTH>16</MAX-LENGTH>
```

```
                                                <MIN-LENGTH>1</MIN-LENGTH>
                                            </DIAG-CODED-TYPE>
                                            <PHYSICAL-TYPE BASE-DATA-
TYPE="A_BYTEFIELD"/>
                                    </DATA-OBJECT-PROP>
                                </DATA-OBJECT-PROPS>
                            </DIAG-DATA-DICTIONARY-SPEC>
                            <DIAG-COMMS>
                                <DIAG-SERVICE ID="DS_11251"
ADDRESSING="FUNCTIONAL-OR-PHYSICAL">
                                    <SHORT-
NAME>DS_ReadSWVersionNumber</SHORT-NAME>
                                    <REQUEST-REF ID-REF="REQ_10471"/>
                                    <POS-RESPONSE-REFS>
                                        <POS-RESPONSE-REF ID-
REF="PR_11131"/>
                                    </POS-RESPONSE-REFS>
                                </DIAG-SERVICE>
                            </DIAG-COMMS>
                            <REQUESTS>
                                <REQUEST ID="REQ_10471">
                                    <SHORT-
NAME>REQ_ReadSWVersionNumber</SHORT-NAME>
                                    <PARAMS>
                                        <PARAM xsi:type="CODED-CONST">
                                            <SHORT-
NAME>PARAM_SID</SHORT-NAME>
                                            <BYTE-POSITION>0</BYTE-
POSITION>
                                            <BIT-POSITION>0</BIT-
POSITION>
                                            <CODED-VALUE>34</CODED-
VALUE>
                                            <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                <BIT-LENGTH>8</BIT-
LENGTH>
                                            </DIAG-CODED-TYPE>
                                        </PARAM>
                                        <PARAM xsi:type="CODED-CONST">
                                            <SHORT-
NAME>DID_SWVersionNumber</SHORT-NAME>
                                            <BYTE-POSITION>1</BYTE-
POSITION>
                                            <BIT-POSITION>0</BIT-
POSITION>
                                            <CODED-VALUE>18193</CODED-
VALUE>
                                            <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                <BIT-LENGTH>16</BIT-
LENGTH>
                                            </DIAG-CODED-TYPE>
                                        </PARAM>
                                    </PARAMS>
                                </REQUEST>
                            </REQUESTS>
                            <POS-RESPONSES>
                                <POS-RESPONSE ID="PR_11131">
                                    <SHORT-
NAME>PR_ReadDataByIdentifier</SHORT-NAME>
                                    <PARAMS>
                                        <PARAM xsi:type="CODED-CONST">
                                            <SHORT-
NAME>PARAM_SID</SHORT-NAME>
                                            <BYTE-POSITION>0</BYTE-
POSITION>
```

```
                                                        <BIT-POSITION>0</BIT-
POSITION>
                                                        <CODED-VALUE>98</CODED-
VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                                  <PARAM xsi:type="CODED-CONST">
                                                        <SHORT-
NAME>DID_SWVersionNumber</SHORT-NAME>
                                                        <BYTE-POSITION>1</BYTE-
POSITION>
                                                        <BIT-POSITION>0</BIT-
POSITION>
                                                        <CODED-VALUE>18193</CODED-
VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>16</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                                  <PARAM xsi:type="VALUE">
                                                        <SHORT-
NAME>Data_SWVersionNumber</SHORT-NAME>
                                                        <BYTE-POSITION>3</BYTE-
POSITION>
                                                        <BIT-POSITION>0</BIT-
POSITION>
                                                        <DOP-SNREF SHORT-
NAME="HexByteField"/>
                                                  </PARAM>
                                            </PARAMS>
                                      </POS-RESPONSE>
                               </POS-RESPONSES>
                        </FUNCTIONAL-GROUP>
                  </FUNCTIONAL-GROUPS>
                  <BASE-VARIANTS>
                        <BASE-VARIANT ID="BV_11271">
                              <SHORT-NAME>BV1</SHORT-NAME>
                              <BASE-VARIANT-PATTERN>
                                    <MATCHING-BASE-VARIANT-PARAMETERS>
                                          <MATCHING-BASE-VARIANT-PARAMETER>
                                                <EXPECTED-
VALUE>AD47110815</EXPECTED-VALUE>
                                                <USE-PHYSICAL-
ADDRESSING>false</USE-PHYSICAL-ADDRESSING>
                                                <DIAG-COMM-SNREF SHORT-
NAME="DS_ReadSWVersionNumber"/>
                                                <OUT-PARAM-IF-SNREF SHORT-
NAME="Data_SWVersionNumber"/>
                                          </MATCHING-BASE-VARIANT-PARAMETER>
                                    </MATCHING-BASE-VARIANT-PARAMETERS>
                              </BASE-VARIANT-PATTERN>
                              <PARENT-REFS>
                                    <PARENT-REF xsi:type="FUNCTIONAL-GROUP-REF" ID-
REF="FG_10424"/>
                              </PARENT-REFS>
                        </BASE-VARIANT>
                        <BASE-VARIANT ID="BV_11331">
                              <SHORT-NAME>BV2</SHORT-NAME>
                              <BASE-VARIANT-PATTERN>
                                    <MATCHING-BASE-VARIANT-PARAMETERS>
                                          <MATCHING-BASE-VARIANT-PARAMETER>
```

```
                                                <EXPECTED-VALUE/>
                                                <DIAG-COMM-SNREF SHORT-
NAME="DS_ReadSWVersionNumber"/>
                                                <OUT-PARAM-IF-SNREF SHORT-
NAME="Data_SWVersionNumber"/>
                                          </MATCHING-BASE-VARIANT-PARAMETER>
                                    </MATCHING-BASE-VARIANT-PARAMETERS>
                              </BASE-VARIANT-PATTERN>
                              <PARENT-REFS>
                                    <PARENT-REF xsi:type="FUNCTIONAL-GROUP-REF" ID-
REF="FG_10424"/>
                              </PARENT-REFS>
                        </BASE-VARIANT>
                  </BASE-VARIANTS>
            </DIAG-LAYER-CONTAINER>
</ODX>
```

The VEHICLE-INFORMATION shown below, now groups both BASE-VARIANTs into an ECU-GROUP. The corresponding GROUP-MEMBERs reference the BASE-VARIANTs and their resolution link. For base variant BV1 a functional and physical resolution link is given, while for base variant BV2 only a physical resolution link exists. Please note that through mixing MATCHING-BASE-VARIANT-PARAMETERs with different values for the element USE-PHYSICAL-ADDRESSING in one BASE-VARIANT-PATTERN, both kinds of links may be needed to resolve the base variant. Functional resolution shall only be used if all BASE-VARIANTs in the ECU-GROUP respond with the same physical address. Otherwise the association of responses to BASE-VARIANTs cannot be performed.

```
<?xml version="1.0" encoding="UTF-8"?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="S:\allgemein\Workgroup\Asam\ASAM_MCD_2D\odx-
Repository\Subversion\odx\dtd\odx.xsd" MODEL-VERSION="2.1.0">
      <VEHICLE-INFO-SPEC ID="VIS_10401">
            <SHORT-NAME>VIS</SHORT-NAME>
            <VEHICLE-INFORMATIONS>
                  <VEHICLE-INFORMATION>
                        <SHORT-NAME>VI1</SHORT-NAME>
                        <LOGICAL-LINKS>
                              <LOGICAL-LINK ID="LL_11481">
                                    <SHORT-NAME>LL_FG</SHORT-NAME>
                                    <PHYSICAL-VEHICLE-LINK-REF ID-
REF="PVL_11511"/>
                                    <FUNCTIONAL-GROUP-REF ID-REF="FG_10424"
DOCREF="DLC_A" DOCTYPE="CONTAINER"/>
                              </LOGICAL-LINK>
                              <LOGICAL-LINK ID="LL_11521">
                                    <SHORT-NAME>LL_BV1</SHORT-NAME>
                                    <PHYSICAL-VEHICLE-LINK-REF ID-
REF="PVL_11511"/>
                                    <BASE-VARIANT-REF ID-REF="BV_11271"
DOCREF="DLC_A" DOCTYPE="CONTAINER"/>
                              </LOGICAL-LINK>
                              <LOGICAL-LINK ID="LL_145261">
                                    <SHORT-NAME>LL_BV2</SHORT-NAME>
                                    <PHYSICAL-VEHICLE-LINK-REF ID-
REF="PVL_11511"/>
                                    <BASE-VARIANT-REF ID-REF="BV_11331"
DOCREF="DLC_A" DOCTYPE="CONTAINER"/>
                              </LOGICAL-LINK>
                        </LOGICAL-LINKS>
                        <ECU-GROUPS>
                              <ECU-GROUP>
                                    <SHORT-NAME>ECUG_SAMPLE</SHORT-NAME>
                                    <GROUP-MEMBERS>
                                          <GROUP-MEMBER>
```

```
                                                        <BASE-VARIANT-REF ID-
REF="BV_11331" DOCREF="DLC_A" DOCTYPE="CONTAINER"/>
                                                        <PHYS-RESOLUTION-LINK-REF
ID-REF="LL_145261"/>
                                                </GROUP-MEMBER>
                                                <GROUP-MEMBER>
                                                        <BASE-VARIANT-REF ID-
REF="BV_11271" DOCREF="DLC_A" DOCTYPE="CONTAINER"/>
                                                        <FUNCT-RESOLUTION-LINK-REF
ID-REF="LL_11481"/>
                                                        <PHYS-RESOLUTION-LINK-REF
ID-REF="LL_11521"/>
                                                </GROUP-MEMBER>
                                        </GROUP-MEMBERS>
                                </ECU-GROUP>
                        </ECU-GROUPS>
                        <PHYSICAL-VEHICLE-LINKS>
                                <PHYSICAL-VEHICLE-LINK ID="PVL_11511"
TYPE="ISO_11898_2_DWCAN">
                                        <SHORT-NAME>PVL_A</SHORT-NAME>
                                        <VEHICLE-CONNECTOR-PIN-REFS>
                                                <VEHICLE-CONNECTOR-PIN-REF ID-
REF="VCPR_0815"/>
                                        </VEHICLE-CONNECTOR-PIN-REFS>
                                </PHYSICAL-VEHICLE-LINK>
                        </PHYSICAL-VEHICLE-LINKS>
                </VEHICLE-INFORMATION>
        </VEHICLE-INFORMATIONS>
</VEHICLE-INFO-SPEC>
</ODX>
```

In the example given resolution of BV1 would proceed as follows:

1. Open Logical Link on functional group level (LL_FG).

2. Execute DIAG-SERVICE DS_ReadSWVersionNumber functionally

3. Compare the response parameter against the expected value

BV2 would be resolved as follows:

1. Open Logical Link on physical level (LL_BV2)

2. Execute DIAG-SERVICE DS_ReadSWVersionNumber physically

3. Compare the response parameter against the expected value

7.4.5 DIAGNOSTIC TROUBLE CODE DESCRIPTION

The service ReadDTCByStatus (ServiceId=18h) is used in ISO 14230 to read diagnostic trouble codes (DTCs) from the ECU's memory using a definite status. After that the service ReadStatusOfDTC (ServiceId=17h) can be used to read environment conditions for a single DTC. The complete implementation of both services can be found in Annex C.2.
At first, the service ReadDTCByStatus should be discussed. The following figure gives an overview of the objects used by this service.

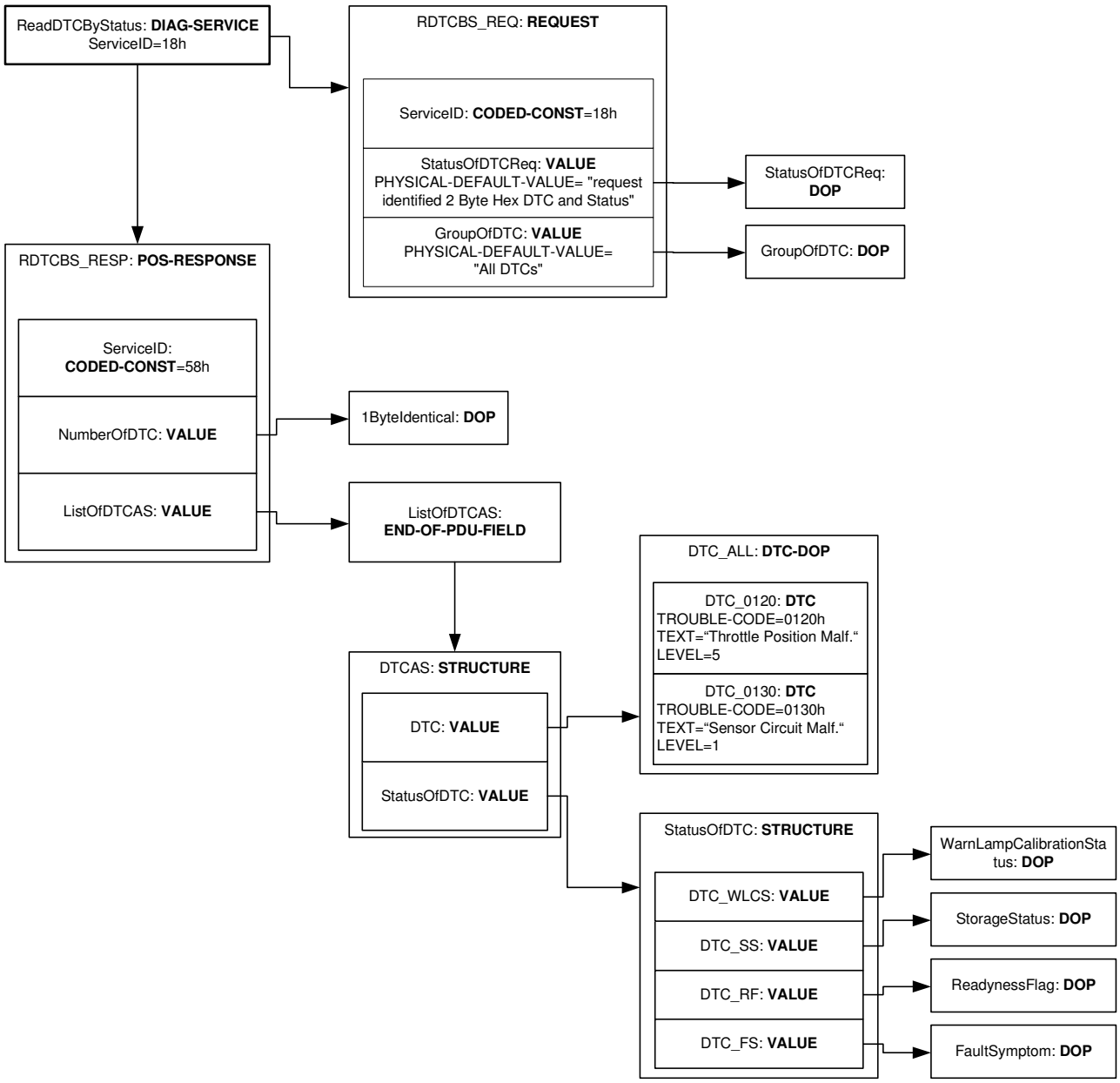


Figure 117 — Objects used by the ReadDTCByStatus service

The service references a request (RDTCBS_REQ) and a positive response (RDTCBS_RESP). The request consists of three parameters: ServiceID, StatusOfDTCReq and GroupOfDTC. The first parameter is of type CODED-CONST and its value is set to 18h. The next two parameters are of type VALUE with predefined default values. Each of them references one DOP (StatusOfDTCReq and GroupOfDTC), which are used for conversion of the values, given by the user, in to their coded representation. The response also includes three parameters: ServiceID as a constant value and NumberOfDTC and ListOfDTC as values, which are to be interpreted. Number of DTCs is computed with a DOP that simply passes a one-byte value on. ListOfDTCAS references an END-OF-PDU-FIELD that interprets the received DTCs as a list of entries. An entry is a structure (DTCAS) consisting of two VALUE parameters: DTC and StatusOfDTC. The DTC parameter references the DTC-DOP with the short-name "DTC_ALL" that is responsible for extraction of the DTC. And at last, the structure StatusOfDTC extracts the

status of DTC consisting of four values: warning lamp calibration status, storage status, readiness flag and fault symptom.
The names of the objects in the Figure 117 correspond to the SHORT-NAMEs of the objects in the XML instance. Here is an example, how the DIAG-SERVICE can be implemented in the XML:

```
<DIAG-SERVICE ID="DS_ReadStatusOfDTC" SEMANTIC="FAULTREAD" ADDRESSING="PHYSICAL">
       <SHORT-NAME>DS_ReadStatusOfDTC</SHORT-NAME>
       <LONG-NAME>Read Status of DTC</LONG-NAME>
       <AUDIENCE IS-DEVELOPMENT="true" IS-AFTERMARKET="true" IS-
MANUFACTURING="true"/>
       <REQUEST-REF ID-REF="REQ_RSODTC"/>
       <POS-RESPONSE-REFS>
              <POS-RESPONSE-REF ID-REF="RESP_RSODTC"/>
       </POS-RESPONSE-REFS>
</DIAG-SERVICE>
```

After all DTCs are read from the ECU's memory, it is possible to query the environmental data, stored with the DTCs. The service ReadStatusOfDTC (ServiceId=17h) is responsible for this task. In the next fugure all objects, used by this service, and relationships between them are shown:

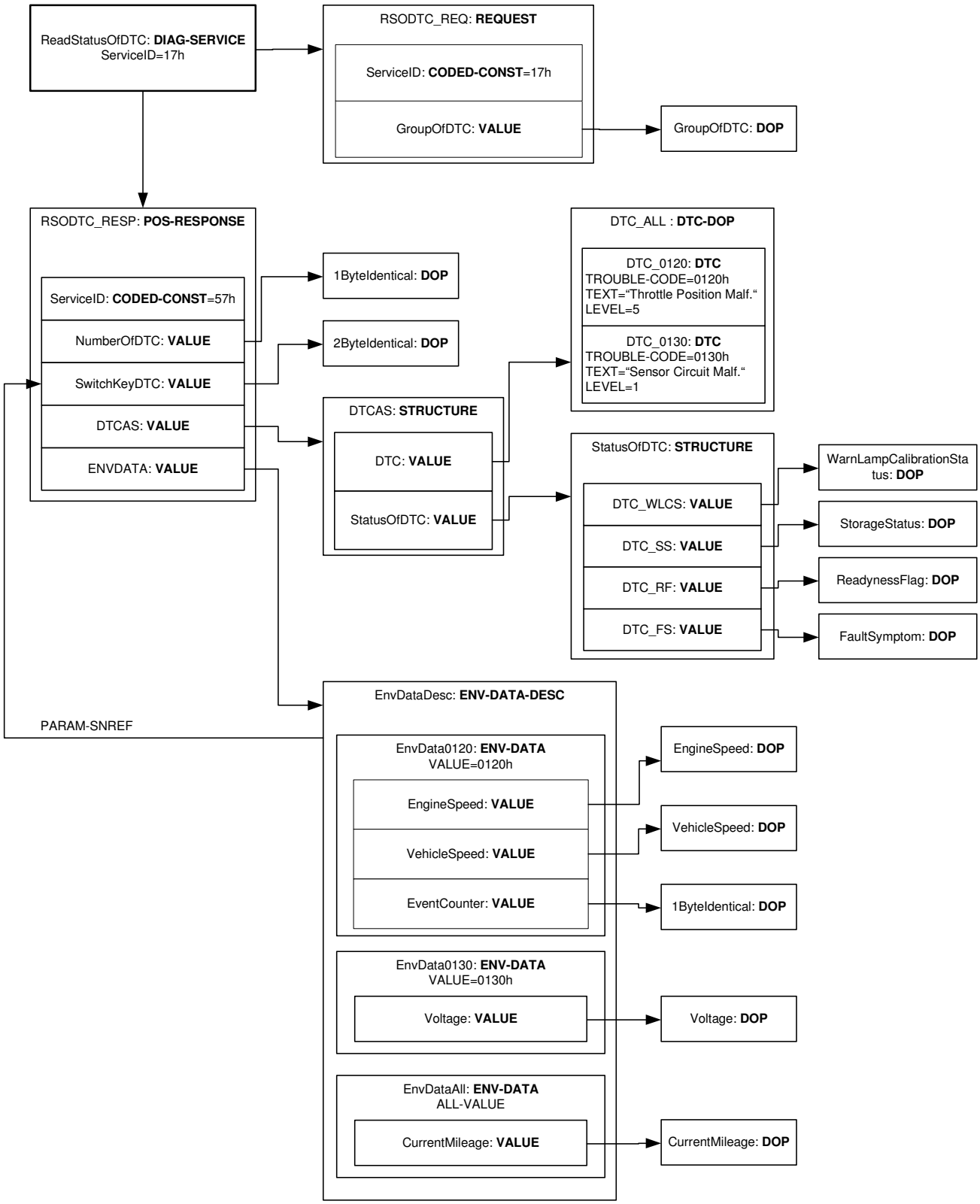


Figure 118 — Objects used by the ReadStatusOfDTC service

The request (RSODTC_REQ), used by this service, contains two parameters: ServiceID and GroupOfDTC. The first one is set to a fixed value of 17h while the user must specify

the second one. The DOP GroupOfDTC converts the user input into the coded value that is put onto the PDU stream.
The response of the service consists of five parameters: ServiceID, NumberOfDTC, DTC, StatusOfDTC and ENVDATA. It is assumed that NumberOfDTC=1 and we only need to interpret a single DTC with its environmental data (this assumtion is also recommended praxis in ISO 14230). The definition of the next two parameters (SwitchKeyDTC and DTCAS) occurs in a tricky way. They both start on the same byte position in the PDU. SwitchKeyDTC parameter is a part of the DTCAS parameter and addresses only to the DTC. In contrast to that, DTCAS references the structure DTCAS including the values DTC and StatusOfDTC. Here is an XML code implementing the response:

```
<POS-RESPONSE ID="RESP_RSODTC">
       <SHORT-NAME>RESP_RSODTC</SHORT-NAME>
       <LONG-NAME>Read Status of DTC Response</LONG-NAME>
       <PARAMS>
              <PARAM xsi:type="CODED-CONST" SEMANTIC="SERVICE-ID">
                     <SHORT-NAME>ServiceID</SHORT-NAME>
                     <LONG-NAME>Read Status of DTC Response</LONG-NAME>
                     <BYTE-POSITION>0</BYTE-POSITION>
                     <CODED-VALUE>87</CODED-VALUE>
                     <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"
xsi:type="STANDARD-LENGTH-TYPE">
                                  <BIT-LENGTH>8</BIT-LENGTH>
                     </DIAG-CODED-TYPE>
              </PARAM>
              <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                     <SHORT-NAME>NumberOfDTC</SHORT-NAME>
                     <LONG-NAME>number of DTC</LONG-NAME>
                     <BYTE-POSITION>1</BYTE-POSITION>
                     <DOP-REF ID-REF="DOP_1ByteIdentical"/>
              </PARAM>
              <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                     <SHORT-NAME>SwitchKeyDTC</SHORT-NAME>
                     <LONG-NAME>DTC SwitchKEY</LONG-NAME>
                     <BYTE-POSITION>2</BYTE-POSITION>
                     <DOP-REF ID-REF="DOP_2ByteIdentical"/>
              </PARAM>
              <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                     <SHORT-NAME>DTCAS</SHORT-NAME>
                     <LONG-NAME>DTC and Status</LONG-NAME>
                     <BYTE-POSITION>2</BYTE-POSITION>
                     <DOP-REF ID-REF="STRUCT_DTCAS"/>
              </PARAM>
              <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                     <SHORT-NAME>ENVDATA</SHORT-NAME>
                     <LONG-NAME>Environmental Data</LONG-NAME>
                     <BYTE-POSITION>5</BYTE-POSITION>
                     <DOP-REF ID-REF="ENVDESC_EnvDataDesc"/>
              </PARAM>
       </PARAMS>
</POS-RESPONSE>
```

The last parameter of the response references the ENV-DATA-DESC object named "EnvDataDesc". It references (via short-name) the parameter SwitchKeyDTC, which is used as switch key to find the right ENV-DATA object within ENV-DATA-DESC. The latter defines three ENV-DATA objects: EnvData0120, EnvData0130 and EnvDataAll. Each of them represents a structure with the appropriate amount of parameters. The last one (EnvDataAll) includes a value (CurrentMileage), which is common for all DTCs, i.e. this value is sent by the ECU as a reply to the request by each DTC (on the BYTE-POSITION=0).
The value DTC is extracted from the PDU twice. First time, it is used as a switch key. Another time, it is a part of the structure DTCAS. Within this structure, the parameter DTC

is extracted as a complex value including trouble code, text and level. The second extraction can be omitted if the DTC should not be shown on display again (it was already shown after reading DTC by status with the service 18h). In this case the fourth parameter would reference the structure SatusOfDTC directly.

7.4.6 PROTOCOL COMMUNICATION PARAMETER

For the exchange of diagnostic data between clients and ECUs it is necessary to assign communication parameters (e.g. timing, etc.) to a specific protocol layer. Only on COMPARAM-SPEC could be referenced from a PROTOCOL layer instance. The communication parameters values defined in the COMPARAM-SPEC could be assigned with new values in the PROTOCOL layer. This festure is necessary if the COMPARAM-SPEC is referenced by more than one PROTOCOL layer.
The following usage scenario will help to understand the handling of the communication-parameters. For that purpose it will be supposed that an examplary ECU supports the KWP2000 protocol on CAN (ISO 14230-3 and ISO 15765-2) for enhanced diagnostic and also the legislated emission related diagnostics on CAN (ISO 15031-5 and ISO 15765-4). Both diagnostic protocols have separate PROTOCOL layer instances and each protocol references the same COMPARAM-SPEC (see the figure below).

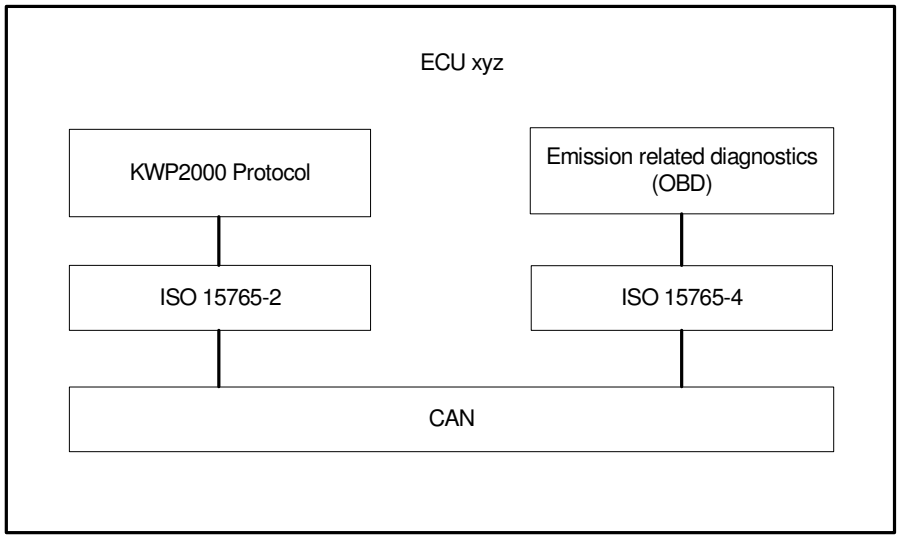


Figure 119 — Diagnostic protocol stack of an exemplary ECU

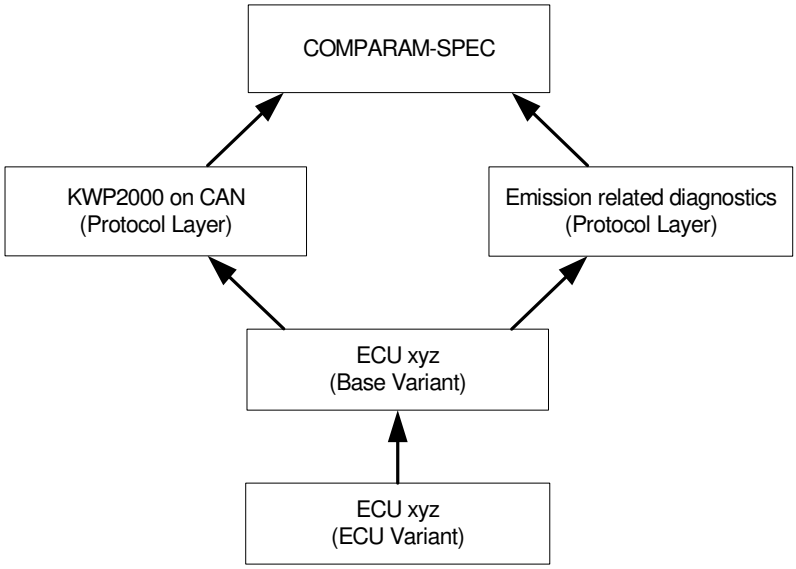


Figure 120 — Simplified Diagnostic layers structure for an exemplary ECU

The common COMPARAM-SPEC instance for both protocols could include the following three communication parameters:

- Address Schema: Select Normal or Extended Addressing
- CANPaddingHandling: Fill unused bytes in every PDU with a special padding bytes
- P2CANServerMax: Maximum Time between Request and Response on ECU side

The COMPARAM-SPEC instance references the COMPARAM-SUBSET instances the parameters are defined in.

```
<COMPARAM-SPEC ID="compKWPonCAN.id123">
    <SHORT-NAME>compKWPonCAN</SHORT-NAME>
       <PROT-STACKS>
         <PROT-STACK ID="compKWPonCAN.PROT-STACK.id1">
              <SHORT-NAME>compKWPonCANprotStack</SHORT-NAME>
              <PDU-PROTOCOL-TYPE>ISO_15031_5_on_ISO_15765_4</PDU-PROTOCOL-TYPE>
              <COMPARAM-SUBSET-REFS>
                <!-- APP -->
                <COMPARAM-SUBSET-REF DOCTYPE="COMPARAM-SUBSET"
DOCREF="compKWPonCANSubset1"
           ID-REF="compKWPonCAN.id1"/>
                <!-- TRANS -->
                <COMPARAM-SUBSET-REF DOCTYPE="COMPARAM-SUBSET"
DOCREF="compKWPonCANSubset2"
           ID-REF="compKWPonCAN.id2"/>
              </COMPARAM-SUBSET-REFS>
         </PROT-STACK>
       </PROT-STACKS>
</COMPARAM-SPEC>
```

The default values for these three communication parameters are initialised with valid values for the Emission related diagnosics. According to the ISO OSI layer model they are defined in several COMPARAM-SUBSET structures.
The ISO 15765-4 Networklayer requires the 'Normal Addressing' as Address Schema.

```
<COMPARAM-SUBSET ID="compKWPonCAN.id1" CATEGORY="APP">
       <SHORT-NAME>compKWPonCAN</SHORT-NAME>
       <COMPARAMS>
```

```
            <COMPARAM CPTYPE="STANDARD" DISPLAY-LEVEL="1"
ID="compKWPonCAN.AddrSchema"                          CPUSAGE="ECU-COMM" PARAM-
CLASS="COM">
                  <SHORT-NAME>AddrSchema</SHORT-NAME>
                  <PHYSICAL-DEFAULT-VALUE>normal</PHYSICAL-DEFAULT-VALUE>
                  <DATA-OBJECT-PROP-REF ID-REF="compKWPonCAN.DOP04"/>
            </COMPARAM>
      </COMPARAMS>
      <DATA-OBJECT-PROPS>...</DATA-OBJECT-PROPS>
</COMPARAM-SUBSET>
```

Each diagnostic CAN frame shall contain eight bytes of data. Therefore CANPaddingHandling shall be enabled. The P2CANServerMax timing for legislated diagnostics is 50 milliseconds.

```
<COMPARAM-SPEC ID="compKWPonCAN.id123">
      <SHORT-NAME>compKWPonCAN</SHORT-NAME>
      <COMPARAMS>
            <COMPARAM CPTYPE="STANDARD" PARAM-CLASS="COM"
ID="compKWPonCAN.AddrSchema"
             DISPLAY-LEVEL="1">
                  <SHORT-NAME>AddrSchema</SHORT-NAME>
                  <PHYSICAL-DEFAULT-VALUE>normal</PHYSICAL-DEFAULT-VALUE>
                  <DATA-OBJECT-PROP-REF ID-REF="compKWPonCAN.DOP04"/>
            </COMPARAM>
            <COMPARAM CPTYPE="OPTIONAL" PARAM-CLASS="COM"
ID="compKWPonCAN.CANPaddingHandling"
             DISPLAY-LEVEL="1">
                  <SHORT-NAME>CANPaddingHandling</SHORT-NAME>
                  <PHYSICAL-DEFAULT-VALUE>enabled</PHYSICAL-DEFAULT-VALUE>
                  <DATA-OBJECT-PROP-REF ID-REF="compKWPonCAN.DOP05"/>
            </COMPARAM>
            <COMPARAM CPTYPE="STANDARD" PARAM-CLASS="TIMING"
ID="compKWPonCAN.P2CANServerMax"
             DISPLAY-LEVEL="1">
                  <SHORT-NAME>P2CANServerMax</SHORT-NAME>
                  <PHYSICAL-DEFAULT-VALUE>50</PHYSICAL-DEFAULT-VALUE>
                  <DATA-OBJECT-PROP-REF ID-REF="compKWPonCAN.DOP06"/>
            </COMPARAM>
      </COMPARAMS>
      <DATA-OBJECT-PROPERTIES>...</DATA-OBJECT-PROPERTIES>
</COMPARAM-SPEC>
```

The protocol layer for emission related diagnostics (OBDonCAN) only needs to reference this COMPARAM-SPEC without any changes.

```
<PROTOCOL ID="protOBDonCAN">
      <SHORT-NAME>protOBDonCAN</SHORT-NAME>
      ...
      <COMPARAM-SPEC-REF ID-REF="compKWPonCAN.id123" DOCREF="compKWPonCAN"
       DOCTYPE="COMPARAM-SPEC"/>
      ...
</PROTOCOL>
```

For enhanced diagnostics (ISO 14230) a different configuration of the communication parameters is be required (extended addressing and no padding of unsed bytes in CAN frames). This makes it necessary to overwrite the different communication parameters in the PROTOCOL layer instance.

```
<PROTOCOL ID="protKWPonCAN">
  <SHORT-NAME>protKWPonCAN</SHORT-NAME>
  ...
  <COMPARAM-SPEC-REF ID-REF="compKWPonCAN.id123" DOCREF="compKWPonCAN"
       DOCTYPE="COMPARAM-SPEC"/>
  ...
  <COMPARAM-REFS>
    <COMPRARM-REF ID-REF="compKWPonCAN.AddrSchema" DOCREF="compKWPonCAN"
```

```
        DOCTYPE="COMPARAM-SPEC">
      <SIMPLE-VALUE>extended</SIMPLE-VALUE>
    </COMPARAM-REF>
    <COMPRARM-REF ID-REF="compKWPonCAN.CANPaddingHandling" DOCREF="compKWPonCAN"
        DOCTYPE="COMPARAM-SPEC">
      <SIMPLE-VALUE>disabled</SIMPLE-VALUE>
    </COMPARAM-REF>
  </COMPARAM-REFS>
  ...
</PROTOCOL>
```

Further on an additional requirement could make it necessary to change the timing of a special DIAG-SERVICE in the ISO 14230 protocol layer.

EXAMPLE The execution of the service 'ClearDiagnosticTroubleCodes' needs 100 millisecond instead of 50 milliseconds defined in the referenced COMPARAM-SPEC because of the time for the EEPROM access. To solve this problem the timing parameter P2CANServerMax is set to the new value of 100ms.

The next example contains all changes for the specific ISO 14230 PROTOCOL layer instance.

```
<PROTOCOL ID="protKWPonCAN.id235">
  <SHORT-NAME>protKWPonCAN</SHORT-NAME>
  ...
   <DIAG-SERVICE ID="ClearDiagTroubleCodes.id456">
   <SHORT-NAME>ClearDiagTroubleCodes</SHORT-NAME>
    <COMPARAM-REFS>
      <COMPARAM-REF ID-REF="compKWPonCAN.P2CANServerMax" DOCREF="compKWPonCAN"
                                                  DOCTYPE="COMPARAM-SPEC">
        <SIMPLE-VALUE>100</SIMPLE-VALUE>
      </COMPARAM-REF>
    <COMPARAM-REFS>
  </DIAG-SERVICE>
 ...
  <COMPARAM-REFS>
    <COMPRARM-REF ID-REF="compKWPonCAN.AddrSchema" DOCREF="compKWPonCAN"
        DOCTYPE="COMPARAM-SPEC">
      <SIMPLE-VALUE>extended</ SIMPLE-VALUE>
    </COMPARAM-REF>
    <COMPRARM-REF ID-REF="compKWPonCAN.CANPaddingHandling" DOCREF="compKWPonCAN"
        DOCTYPE="COMPARAM-SPEC">
      < SIMPLE-VALUE>disabled</ SIMPLE-VALUE>
    </COMPARAM-REF>
  </COMPARAM-REFS>
  ...
  <COMPARAM-SPEC-REF ID-REF="compKWPonCAN.id123" DOCREF="compKWPonCAN"
        DOCTYPE="COMPARAM-SPEC"/>
  ...
  ...
</PROTOCOL>
```

7.4.7 DYNAMIC DIAGNOSTIC RESPONSE

USAGE EXAMPLE OF DYNAMIC-LENGTH-FIELD

The following figure shows the overview of objects used by the response of the service readDTCByStatus (18 h) in ISO 14230.

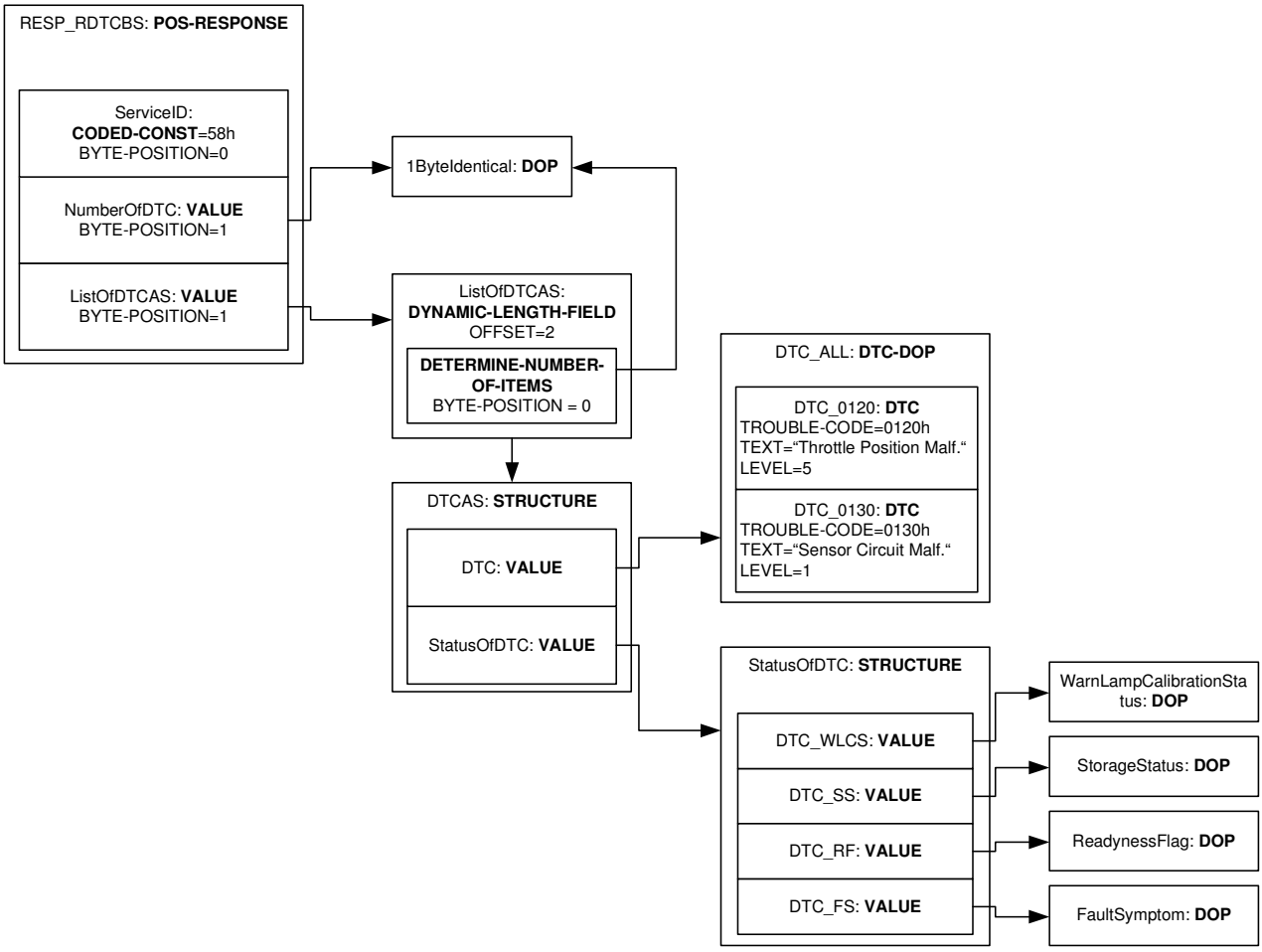


Figure 121 — DYNAMIC-LENGTH-FIELD in readDTCByStatus response

DYNAMIC-LENGTH-FIELD is used there to interpret the list of diagnostic trouble codes (DTCs) received from the ECU. It references the structure DTCAS, which is taken for interpretation of a single DTC with the associated status. The parameter NumberOfDTC specifies how often this structure is repeated. Therefore, the member DETERMINE-NUMBER-OF-ITEMS refers to this parameter by the BYTE-POSITION and to the DOP 1ByteIdentical used to extract the number of DTCs out of the PDU.
In XML language the DYNAMIC-LENGTH-FIELD looks like this:

```
<DYNAMIC-LENGTH-FIELD ID="ListOfDTCAS">
       <SHORT-NAME>ListOfDTCAS</SHORT-NAME>
       <BASIC-STRUCTURE-REF ID-REF="STRUCT_DTCAS"/>
       <OFFSET>1</OFFSET>
       <DETERMINE-NUMBER-OF-ITEMS>
              <BYTE-POSITION>0</BYTE-POSITION>
              <DATA-OBJECT-PROP-REF ID-REF="DOP_1ByteIdentical"/>
       </DETERMINE-NUMBER-OF-ITEMS>
</DYNAMIC-LENGTH-FIELD>
```

In this example the list ListOfDTCAS reaches the end of the PDU, so we actually do not need the number of DTCs and the class END-OF-PDU-FIELD can be used instead of DYNAMIC-LENGTH-FIELD. This alternative implementation can be found in section 7.4.5.

USAGE EXAMPLE OF DYNAMIC-ENDMARKER-FIELD

In the following example a DYNAMIC-ENDMARKER-FIELD is defined, which contains a field of elements with the structure defined by the object with ID="STRUCT-3ByteWrapper". Before each iteration the DOP with ID="DOP-2ByteIdentical" extracts a number from the PDU (a 2-byte value) at the current position which is compared with TERMINATION-VALUE. The iteration is stopped when the value 65535 (FFFFh) is reached.

```
<DYNAMIC-ENDMARKER-FIELD ID="DEMFIELD_3ByteValuesUntilFFFF">
       <SHORT-NAME>3ByteValuesUntilFFFF</SHORT-NAME>
       <BASIC-STRUCTURE-REF ID-REF="STRUCT_3ByteWrapper"/>
       <DATA-OBJECT-PROP-REF ID-REF="DOP_2ByteIdentical">
              <TERMINATION-VALUE>65535</TERMINATION-VALUE>
       </DATA-OBJECT-PROP-REF>
</DYNAMIC-ENDMARKER-FIELD>
```

Assuming, the tester receives the following byte sequence:

```
75 45 32 FF FF EE 1A FF FF
```

Having the following object structure of the response:

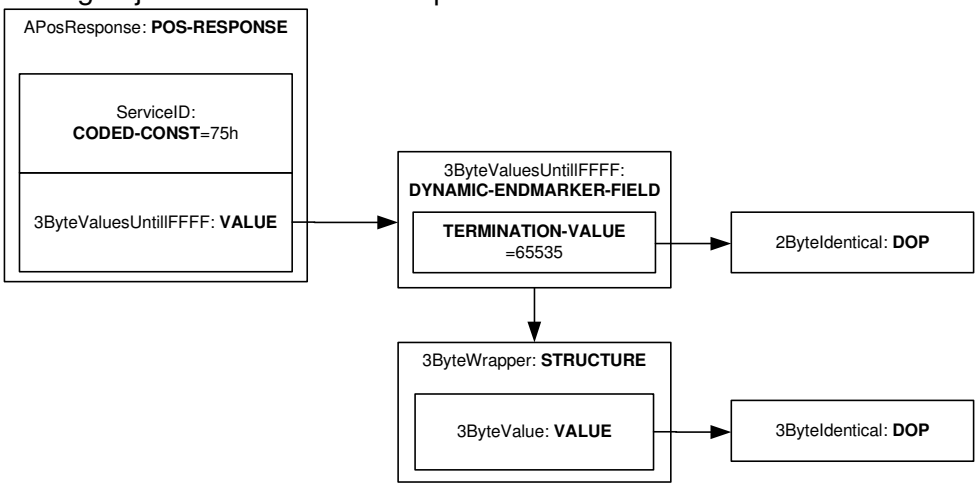


Figure 122 — DYNAMIC-ENDMARKER-FIELD in a response

We get these two 3-byte values: 4532FFh and FFEE1Ah. The first byte of the PDU is the service identifier (75h). Then the value 4532h is extracted from the PDU using the DOP 2ByteIdentical. Since the extracted value does not equal 65535 (FFFFh), the structure 3ByteWrapper is applied. It extracts a 3-byte value (4532FFh) using the DOP 3ByteIdentical. Then the DOP 2ByteIdentical is applied again. FFEEh does not match the termination value, so the structure 3ByteWrapper extracts a 3-byte value (FFEE1Ah) for the second time. Finally, the DOP 2ByteIdentical extracts FFFFh and the iteration stops because the extracted value matches the given termination value. 4532FFh and FFEE1Ah are then presented to the user.
The termination value FFFFh can be found twice in the PDU above. Only the second occurrence of this value is accepted as termination value, because the first occurrence can never be extracted by the DOP 2ByteIdentical due to the unfavourable byte arrangement.

7.4.7.1 USAGE EXAMPLE OF END-OF-PDU-FIELD

See implementation of the response RDTCBS_RESP in section 7.4.5.

7.4.7.2 USAGE EXAMPLE OF MUX

In ISO 14230 the service ReadEcuIdentification (1Ah) is used to read data for ECU identification purposes. By means of this service it is possible for example to request the VIN (Vehicle Identification Number) or the programming date of the software. In the request the parameter identificationOption defines the kind of the requested information. Depending on it, different DOPs are used for the interpretation of the data received from the ECU. The following figure shows how this service can be implemented in ODX:

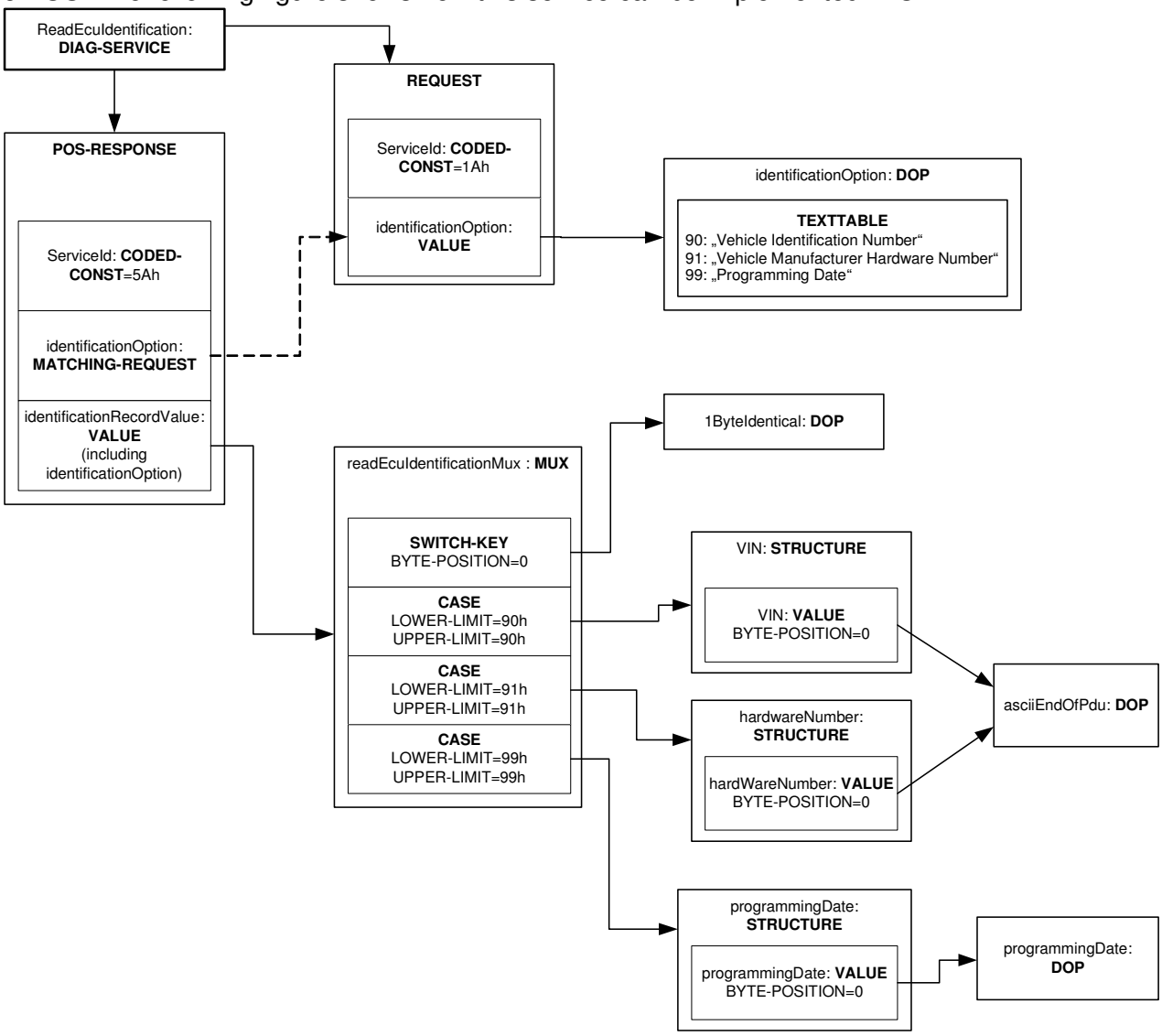


Figure 123 — MUX in ReadEcuIdentification response

The request of the service contains two parameters: ServiceId and identificationOption. In the response the value of identificationOption is repeated after the parameter posRespServiceId (ServiceId + 40h). After that the requested data follow, which is interpreted by the MUX.
The following example code implements the MUX in XML:

```
<MUX ID="MUX_ReadEcuIdentificationMux">
      <SHORT-NAME>MUX_ReadEcuIdentificationMux</SHORT-NAME>
      <BYTE-POSITION>1</BYTE-POSITION>
      <SWITCH-KEY>
            <BYTE-POSITION>0</BYTE-POSITION>
```

```
            <DATA-OBJECT-PROP-REF ID-REF="DOP_1ByteIdentical"/>
        </SWITCH-KEY>
        <CASES>
            <CASE>
                <SHORT-NAME>from90to90hex</SHORT-NAME>
                <STRUCTURE-REF ID-REF="STRUCT_VIN"/>
                <LOWER-LIMIT>144</LOWER-LIMIT>
                <UPPER-LIMIT>144</UPPER-LIMIT>
            </CASE>
            <CASE>
                <SHORT-NAME>from91to91hex</SHORT-NAME>
                <STRUCTURE-REF ID-REF="STRUCT_hardwareNumber"/>
                <LOWER-LIMIT>145</LOWER-LIMIT>
                <UPPER-LIMIT>145</UPPER-LIMIT>
            </CASE>
            <CASE>
                <SHORT-NAME>from99to99hex</SHORT-NAME>
                <STRUCTURE-REF ID-REF="STRUCT_programmingDate"/>
                <LOWER-LIMIT>153</LOWER-LIMIT>
                <UPPER-LIMIT>153</UPPER-LIMIT>
            </CASE>
        </CASES>
</MUX>
```

STRUCT_VIN is used for interpretation of the parameter, which references this MUX, if the byte before this parameter equals 90h. STRUCT_hardwareNumber is used if this byte has a value of 91h and finally, programmingDate is relevant in the case of 99h.

7.4.8 VARIABLE LENGTH PARAMETER

This usage scenario demonstrates the use of the LENGTH-KEY parameter. There are some cases, in which one parameter determines the length of the other one. E.g. the readMemoryByAddress service in ISO 14229 uses a parameter called "addressAndLengthFormatIdentifier" to define the length of the following two parameters called "memoryAddress" and "memorySize". The following figure shows how the request of this service could be implemented in ODX.

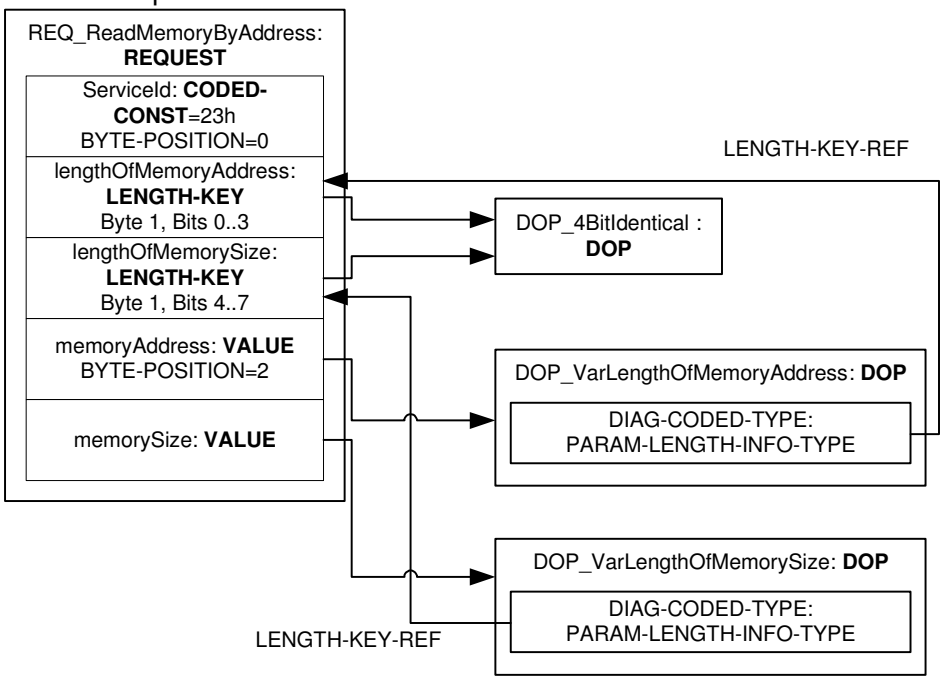


Figure 124 — LENGTH-KEY parameter in request of readMemoryByAddress service

The length of the parameter "memoryAddress" is given by the lower nibble of byte 1 (parameter "lengthOfMemoryAddress"). Therefore the DATA-OBJECT-PROP referenced by this parameter uses PARAM-LENGTH-INFO-TYPE as DIAG-CODED-TYPE. The latter refers to the parameter "lengthOfMemoryAddress" defining the legth of the parameter "memoryAddress". The length of the parameter "memorySize" is defined in the similar way. While the position of the first three parameters can be declared in the ODX instance, the position of the last parameter is unknown until runtime. For that reason the BYTE-POSITION of the last parameter is not defined there.

7.4.9 FUNCTIONAL ADDRESSING

This section describes how data for functional addressing communication scenarios can be defined using ODX in a mostly protocol neutral manner.

7.4.9.1 COMMUNICATION ON THE VEHICLE BUS

Functional Addressing depicts a communication scenario, where a tester sends out a request on a dedicated functional address and multiple ECUs respond to such a request. It closely resembles Broadcast communication in common network technology. In opposition to functional addressing, the term physical addressing is used, if the tester sends a request to an address uniquely identifying one ECU on the bus.
The communication scenario on the vehicle bus is shown in the following figure. A tester sends a functional request on the bus to which any number of ECUs on the bus may be responding. In the scenario shown one ECU does not respond to the functional request.

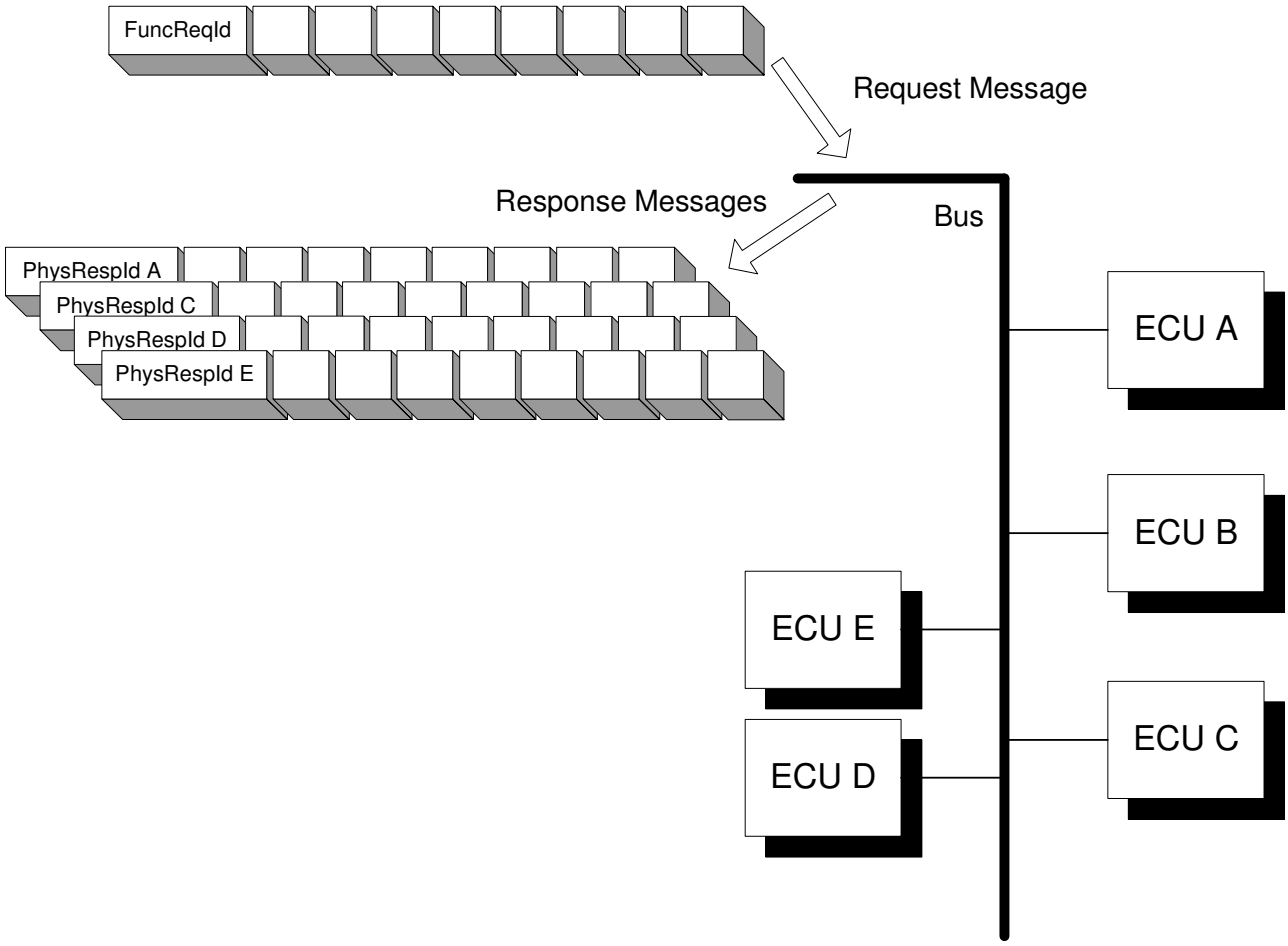


Figure 125 — Functional Addressing Communication on the vehicle bus

In general, at the time of sending out the request, the tester does not know how many responses it will receive. This may have multiple reasons:

- An ECU is broken and does not respond even though it should
- An ECU listed in the vehicle information is not really built into the particular vehicle under test

By consequence, the D-PDU-API and its protocol drivers do not know when the communication primitive initiated through the request message terminates. It is assumed in the following that termination of a functionally addressed communication primitive is always detected on a time-out. In these cases, the time-out does not indicate an error or exceptional situation and shall not be handled as such by either the D-PDU-API or the MVCI Server.

Another important observation is that not all ECU's responses may have the exactly same structure. Even when by definition they should, mis-implementations of the diagnostic protocol in ECUs also have to be considered.

Furthermore, communication scenarios exist, where the ECU sends a multi-part response, that is multiple individual response messages to one single request, where even the individual responses may not be structured identically. This makes for the most complicated functional addressing scenario, as the multiple responses of multiple ECUs have to be assigned to the right physical ECUs correctly.

It also needs to be considered that ECUs do not necessarily respond to the functional request using a physical response identifier. Communication scenarios exist, where the ECU will respond using a functional response identifier.

7.4.9.2 DATA STRUCTURES IN ODX TO DEFINE FUNCTIONAL ADDRESSING

To define functional addressing communication in ODX, the following aspects need to be considered and will be explained in detail:

- Definition of the functional address for the request message
- Definition of the list of response identifiers for all possibly responding ECUs
- Definition of the functional DIAG-SERVICE(s) within a FUNCTIONAL-GROUP
- Definition of possibly overridden RESPONSE messages for some of the responding ECUs

The definition of data for functional addressing communication scenarios shall permit a runtime system to open a logical link on a FUNCTIONAL-GROUP and communicate functionally without having to consider any BASE-VARIANT data. This is important for e.g. the OBD use case. However, if BASE-VARIANT data is available, functional communication should also be able to use this data.
The most important element for a functional addressing scenario is the FUNCTIONAL-GROUP. Only DIAG-SERVICES contained in a functional group and with addressing modes FUNCTIONAL or FUNCTIONAL-OR-PHYSICAL can be executed functionally within an MVCI Server. For this the application has to open a logical link on such a FUNCTIONAL-GROUP.
The functional request address needs to be supported by the protocol. That is, the FUNCTIONAL-GROUP must inherit from a PROTOCOL layer and within this PROTOCOL layer's COMPARAM-SPEC one or multiple communication parameters for the functional request address need to be defined. The value of this or these communication parameter(s) can then be re-defined on the level of the FUNCTIONAL-GROUP.
A similar statement holds true for the list of response identifiers or response addresses for the possibly responding ECUs. Here, too, the COMPARAM-SPEC of the protocol layer needs to support a COMPLEX-COMPARAM of CPTYPE UNIQUE_ID to set up this address table. The content of the COMPLEX-COMPARAM can then again be overridden on the FUNCTIONAL-GROUP level. The following constraints hold on the setup of this COMPLEX-COMPARAM:

1. For every protocol its SHORT-NAME must be *CP_UniqueRespIdTable*. This is important, because the MVCI server needs to be able to unambiguously find this COMPARAM to correctly feed the D-PDU-API interface.
2. The ALLOW_MULTIPLE_VALUES attribute of the *CP_UniqueRespIdTable* must be set to "true".
3. The general content of this COMPLEX-COMPARAM must be filled in a protocol-specific manner. The details for standardized protocols can be found in in the D-PDU-API specficiation. For non-standardized protocols this needs to be defined amongst the users of this protocol.
4. This COMPLEX-COMPARAM always shall contain a COMPARAM with SHORT-NAME *CP_ECULayerShortName*. This communication parameter too is needed by the MVCI Server to set up the correct Access Keys for every received response. The CPUSAGE of this communication parameter must be set to APPLICATION.
5. The CPUSAGE of the COMPLEX-COMPARAM and all contained COMPARAMs except the one listed in item 3 shall be set to UNIQUE_ID.

To enable functional addressing in the D-PDU-API the defined communication parameter CP_AddressMode must be set to functional addressing on the FUNCTIONAL-GROUP level and to physical addressing on any lower layer.

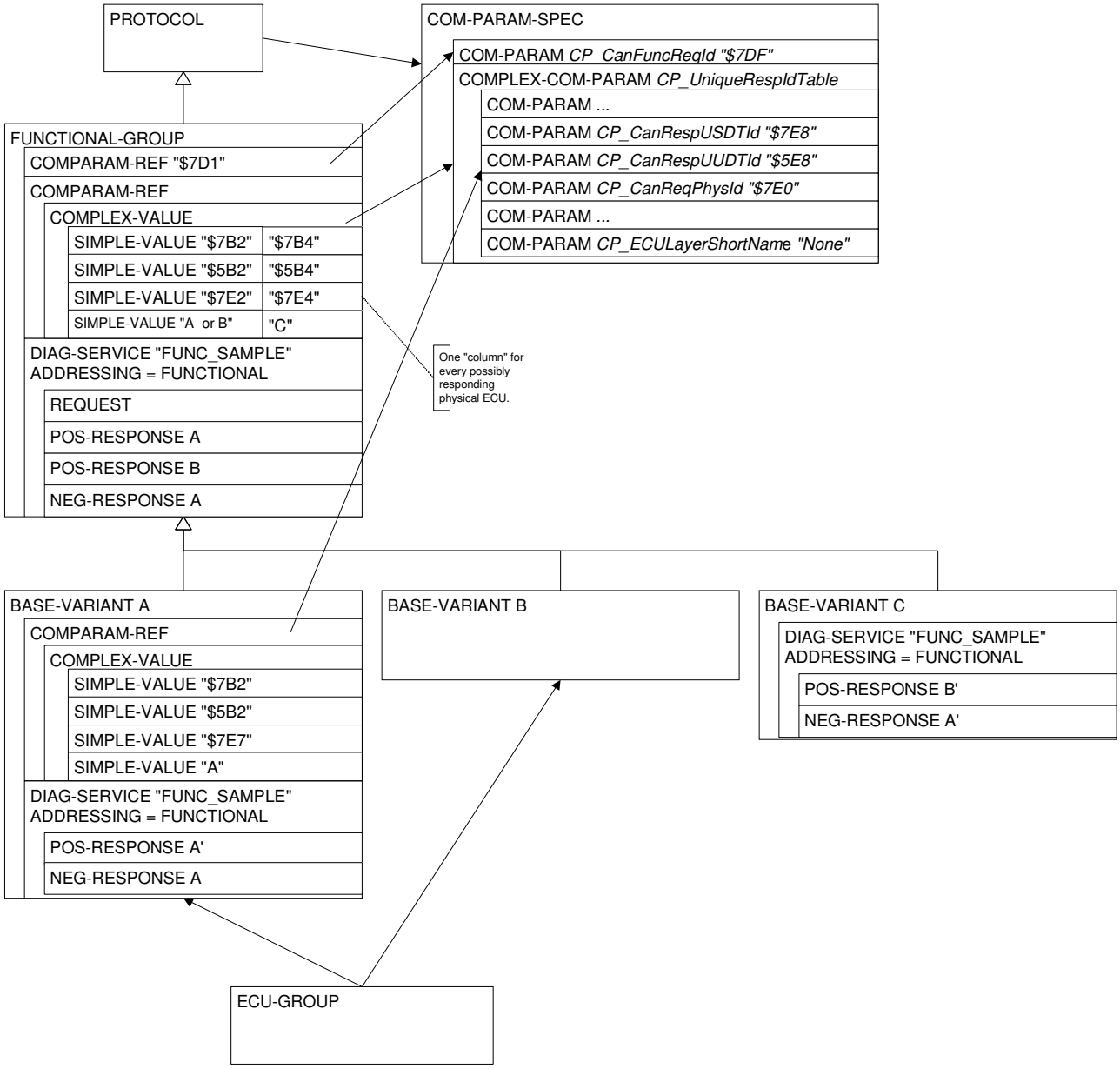


Figure 126 — Functional Addressing Setup of ODX Data

Overridden RESPONSE messages are defined on the level of BASE-VARIANTs in the regular manner: A DIAG-SERVICE with the same SHORT-NAME as on the FUNCTIONAL-GROUP layer is introduced. A new RESPONSE is referenced by this overridden service. For overridden response messages the relevance of ECU-GROUPs needs to be analyzed. In case multiple BASE-VARIANTs belonging to the same ECU-GROUP are also part of the same FUNCTIONAL-GROUP, consideration of overridden RESPONSE messages of such BASE-VARIANTs requires a previous BASE-VARIANT identification step.

The overall composition of data structures to set up functional communication scenarios is depicted in the figure above. Please note that this figure shows the most complex setup. Easier setups without involvement of ECU-GROUPs or even without involvement of BASE-VARIANTs also exist. The shown setup has a FUNCTIONAL-GROUP holding a *CP_UniqueRespIdTable* with entries for 2 possibly responding ECUs. Three BASE-VARIANTs inherit from the FUNCTIONAL-GROUP. BASE-VARIANT A overrides the complex communication parameter value for *CP_UniqueRespIdTable* and also overrides

the functional service. BASE-VARIANT B neither overrides the communication parameter nor the functional service. BASE-VARIANT C overrides the service only. BASE-VARIANTs A and B reside in an ECU-GROUP. By consequence, only one of the two can actually be built into any given vehicle.

7.4.9.3 Sequence of Events for Functional Addressing

The following describes a sequence of events for interpreting the ODX data structures correctly in a functional addressing communication scenario.

1. Application opens a logical link on FUNCTIONAL-GROUP level via MVCI server.
2. Application selects DIAG-SERVICE with ADDRESSING = FUNCTIONAL | FUNCTIONAL-OR-PHYSICAL and executes it.
3. MVCI Server composes the functional request
    a. It sets up a runtime RespId Table including all the necessary information to map responses to their physical source ECUs.
        i. It takes the COMPLEX-COMPARAM *CP_UniqueRespIdTable* with the values as defined on the FUNCTIONAL-GROUP level. (Important: The communication parameter *CP_ECULayerShortName* is to be used only within the MVCI server. It shall not be sent to the D-PDU-API).
        ii. For every BASE-VARIANT that is not part of an ECU-GROUP add the overridden *CP_UniqueRespIdTable* values to the one created in step i. If entries of CP_ECULayerShortName are duplicate, the entry found on BASE-VARIANT level has higher priority than the one on FUNCTIONAL-GROUP level. The entries found on BASE-VARIANT level shall always be added before the ones on FUNCTIONAL-GROUP level. (With this setup the D-PDU-API can go through the table in the right order of priorities)
        iii. For every BASE-VARIANT that is part of an ECU-GROUP but has been identified through a base variant identification process add the overridden *CP_UniqueRespIdTable* values to the one created in step ii. If entries are duplicate, the entry found on BASE-VARIANT level has higher priority than the one on FUNCTIONAL-GROUP level.
        iv. For every BASE-VARIANT that is part of an ECU-GROUP and has not yet been identified through a base variant identification process, ignore any possibly overridden *CP_UniqueRespIdTable* values.
        v. For every entry in the resulting runtime RespId Table add a unique identifier. This identifier will be returned by the D-PDU-API, if a response was mapped to this entry in the table (this unambiguously identifies the source of the response message).
    b. It sets up the expected response table (table with acceptance masks for every possible received response message) information for every response that can possibly be received.
        i. Acceptance mask information is set up per communication primitive but for the complete functional group. That is, POS- and NEG-RESPONSE data from FUNCTIONAL-GROUP level and BASE-VARIANT level are combined into one expected response table.
        ii. In case a FUNCTIONAL-GROUP has no inheriting BASE-VARIANTs the possible physical sources are determined through the *CP_UniqueRespIdTable* communication parameter value. The expected response table (the acceptance masks) are set up using all POS-RESPONSEs and NEG-RESPONSEs as listed for the DIAG-SERVICE on the FUNCTIONAL-GROUP level.

iii. In case a FUNCTIONAL-GROUP has inheriting BASE-VARIANTs, the possible physical sources are the union of entries in the *CP_UniqueRespIdTable* and the inheriting BASE-VARIANTs.
   1. For every entry in the *CP_UniqueRespIdTable* that does not match an inherited BASE-VARIANT with respect to the value of *CP_UniqueRespIdTable*, add the POS-RESPONSE and NEG-RESPONSE acceptance mask data to the expected response table.
   2. If the entry in the *CP_UniqueRespIdTable* does match with an inherited BASE-VARIANT and that BASE-VARIANT does not reside in an ECU-GROUP add the BASE-VARIANT's overridden POS-RESPONSEs and NEG-RESPONSEs to the expected response table for the DIAG-SERVICE .
   3. If the entry in the *CP_UniqueRespIdTable* does match with an inherited BASE-VARIANT and that BASE-VARIANT does reside in an ECU-GROUP but the BASE-VARIANT has been identified through a base variant identification process follow the guidelines in 2 above.
   4. If the entry in the *CP_UniqueRespIdTable* does match with an inherited BASE-VARIANT and that BASE-VARIANT does reside in an ECU-GROUP but the BASE-VARIANT has not yet been identified through a base variant identification process follow the guidelines in 1 above.
   5. (Remark: Through this process multiple identical acceptance masks could be set up in the expected response table. The MVCI server has to ensure uniqueness. In case mutliple equivalent acceptance masks would have to be added that really represent different messages (e.g. one on the FUNCTIONAL-GROUP level and one on the BASE-VARIANT level), the MVCI server needs to store internally, that a D-PDU-API returned acceptance Id points to different response message templates dependent on the UNIQUE-ID that has previously been obtained – see below).

c. It uses the functional request address communication parameter(s) to fill the header information.

4. MVCI Server sends out the functional request.
5. D-PDU-API collects all incoming responses until a timeout occurs. Every response is sent directly to the MVCI server. This is also true for multi-part responses (e.g. UUDT messages with ISO 15765-2 based protocols)
6. MVCI Server matches the unique identifier returned with every response message to identify the physical source of the response message. It also takes the returned acceptance id to select the RESPONSE template to decode the response PDU into physical data.
7. MVCI Server sets the Access Key information for every received response based on the UNIQUE-ID and acceptance id returned from the D-PDU-API.
   a. In case the response was matched against a response template as defined on the FUNCTIONAL-GROUP level (cases b.iii.1. and b.iii.4.) build the Access Key as follows: <PROTOCOL.SHORT-NAME>.<FUNCTIONAL-GROUP.SHORT-NAME>.<*CP_ECULayerShortName*.SIMPLE-VALUE>
   b. In case the response was matched against the overridden response template as defined on a BASE-VARIANT level (cases b.iii.2. and b.iii.3) and the CP_ECULayerShortName parameter value has been overridden on BASE-VARIANT level build the Access Key as follows:

<PROTOCOL.SHORT-NAME>.<FUNCTIONAL-GROUP.SHORT-NAME>.>.<*CP_ECULayerShortName*.SIMPLE-VALUE>

c. In case the response was matched against the overridden response template as defined on a BASE-VARIANT level (cases b.iii.2. and b.iii.3) and the CP_ECULayerShortName parameter value has not been overridden on BASE-VARIANT level build the Access Key as follows: <PROTOCOL.SHORT-NAME>.<FUNCTIONAL-GROUP.SHORT-NAME>.<BASE-VARIANT.SHORT-NAME>

8. MVCI Server sends responses to application within one Result as soon as the communication primitive was terminated by the D-PDU-API, which requires occurrence of a timeout.
9. Application interprets result as necessary.

For the example structure given in Figure 126 the sequence of events is as follows:

Case 1: No BASE-VARIANT identification performed

In case no BASE-VARIANT identification has been performed yet the sequence of events is:

3.a.i) The RespId Table set up is:

Resp Id (Built by the MVCI server at runtime)	COMPARAM 1	COMPARAM 2	COMPARAM 3	*CP_ECULayerShortName*
1	$7B2	$5B2	$7E2	“A or B”
2	$7B4	$5B4	$7E4	“C”

3.b.) The acceptance masks are set up from the following responses

- For physical source “A or B”: POS-RESPONSES A&B, NEG-RESPONSE A
- For physical source “C”: POS-RESPONSE B’, NEG-RESPONSE A’

7) The Access-Keys would be set-up as follows:

- For responses on RespID 1: Use *CP_ECULayerShortName*
- For responses on RespID 2: Use BASE-VARIANT C SHORT-NAME

Case 2: BASE-VARIANT identification performed, identified BASE-VARIANT A

In case BASE-VARIANT A has been identified the sequence of events is:

3.a.i) The RespId Table set up is:

1	$7B2	$5B2	$7E7	“A”
2	$7B4	$5B4	$7E4	“C”

3.b.) The acceptance masks are set up from the following responses

- For physical source “A”: POS-RESPONSE A’, NEG-RESPONSE A
- For physical source “C”: POS-RESPONSE B’, NEG-RESPONSE A’

7) The Access-Keys would be set-up as follows:

- For responses on RespID 1: Use *CP_ECULayerShortName*
- For responses on RespID 2: Use BASE-VARIANT C SHORT-NAME

Case 3: BASE-VARIANT identification performed, identified BASE-VARIANT B

In case BASE-VARIANT B has been identified the sequence of events is:

3.a.i) The RespId Table set up is:

1	$7B2	$5B2	$7E2	“A or B”
2	$7B4	$5B4	$7E4	“C”

(Remark: “A or B” remains in the CP_ECULayerShortName COMPARAM, because the

3.b.) The acceptance masks are set up from the following responses
- For physical source “B”: POS-RESPONSEs A&B, NEG-RESPONSE A
- For physical source “C”: POS-RESPONSE B’, NEG-RESPONSE A’

7) The Access-Keys would be set-up as follows:
- For responses on RespID 1: Use BASE-VARIANT B SHORT-NAME (Important: The “A or B” setting for *CP_ECULayerShortName* is ignored in this case)
- For responses on RespID 2: Use BASE-VARIANT C SHORT-NAME

7.4.9.4 ODX EXAMPLES

In the following, an ODX example is given that reflects the structure in the figure above. A COMPARAM-SUBSET and corresonponding COMPARAM-SPEC are defined in the following way (reduced view, these are not complete communication parameter specifications for the protocol given):

```
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
      <COMPARAM-SUBSET ID="CP_SUBS_11361" CATEGORY="TRANSPORT">
            <SHORT-NAME>CPS_UDS_on_CAN</SHORT-NAME>
            <COMPARAMS>
                  <COMPARAM ID="CP_14291" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_CanFuncReqId</SHORT-NAME>
                        <PHYSICAL-DEFAULT-VALUE>2015</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP_11482"/>
                  </COMPARAM>
            </COMPARAMS>
            <COMPLEX-COMPARAMS>
                  <COMPLEX-COMPARAM ID="CCP_11381" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM" ALLOW-MULTIPLE-
VALUES="true">
                        <SHORT-NAME>CP_UniqueRespIdTable</SHORT-NAME>
                        <COMPARAM ID="CP_11401" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_CanPhysReqFormat</SHORT-NAME>
                              <PHYSICAL-DEFAULT-VALUE>4</PHYSICAL-DEFAULT-
VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP_11441"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_11461" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_CanPhysReqId</SHORT-NAME>
                              <PHYSICAL-DEFAULT-VALUE>2016</PHYSICAL-DEFAULT-
VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP_11482"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_11531" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_CanPhysReqExtAddr</SHORT-NAME>
                              <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-
VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP_11441"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_11551" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_CanRespUSDTFormat</SHORT-NAME>
                              <PHYSICAL-DEFAULT-VALUE>4</PHYSICAL-DEFAULT-
VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP_11441"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_11571" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_CanRespUSDTId</SHORT-NAME>
```

```
                                    <PHYSICAL-DEFAULT-VALUE>2024</PHYSICAL-DEFAULT-
VALUE>
                                    <DATA-OBJECT-PROP-REF ID-REF="DOP_11482"/>
                             </COMPARAM>
                             <COMPARAM ID="CP_11581" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                                    <SHORT-NAME>CP_CanRespUSDTExtAddr</SHORT-NAME>
                                    <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-
VALUE>
                                    <DATA-OBJECT-PROP-REF ID-REF="DOP_11441"/>
                             </COMPARAM>
                             <COMPARAM ID="CP_11591" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                                    <SHORT-NAME>CP_CanRespUUDTFormat</SHORT-NAME>
                                    <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-
VALUE>
                                    <DATA-OBJECT-PROP-REF ID-REF="DOP_11441"/>
                             </COMPARAM>
                             <COMPARAM ID="CP_12001" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                                    <SHORT-NAME>CP_CanRespUUDTExtAddr</SHORT-NAME>
                                    <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-
VALUE>
                                    <DATA-OBJECT-PROP-REF ID-REF="DOP_11441"/>
                             </COMPARAM>
                             <COMPARAM ID="CP_12021" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                                    <SHORT-NAME>CP_CanRespUUDTId</SHORT-NAME>
                                    <PHYSICAL-DEFAULT-VALUE>1512</PHYSICAL-DEFAULT-
VALUE>
                                    <DATA-OBJECT-PROP-REF ID-REF="DOP_11482"/>
                             </COMPARAM>
                             <COMPARAM ID="CP_14261" PARAM-CLASS="UNIQUE_ID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="APPLICATION">
                                    <SHORT-NAME>CP_ECULayerShortName</SHORT-NAME>
                                    <PHYSICAL-DEFAULT-VALUE>"None"</PHYSICAL-
DEFAULT-VALUE>
                                    <DATA-OBJECT-PROP-REF ID-REF=""/>
                             </COMPARAM>
                      </COMPLEX-COMPARAM>
               </COMPLEX-COMPARAMS>
               <DATA-OBJECT-PROPS>
                      <DATA-OBJECT-PROP ID="DOP_11441">
                             <SHORT-NAME>One_byte_identical</SHORT-NAME>
                             <COMPU-METHOD>
                                    <CATEGORY>IDENTICAL</CATEGORY>
                             </COMPU-METHOD>
                             <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"
xsi:type="STANDARD-LENGTH-TYPE">
                                    <BIT-LENGTH>8</BIT-LENGTH>
                             </DIAG-CODED-TYPE>
                             <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
                      </DATA-OBJECT-PROP>
                      <DATA-OBJECT-PROP ID="DOP_11482">
                             <SHORT-NAME>Four_byte_identical</SHORT-NAME>
                             <COMPU-METHOD>
                                    <CATEGORY>IDENTICAL</CATEGORY>
                             </COMPU-METHOD>
                             <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"
xsi:type="STANDARD-LENGTH-TYPE">
                                    <BIT-LENGTH>32</BIT-LENGTH>
                             </DIAG-CODED-TYPE>
                             <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
                      </DATA-OBJECT-PROP>
                      <DATA-OBJECT-PROP ID="DOP_14272">
                             <SHORT-NAME>String</SHORT-NAME>
                             <COMPU-METHOD>
```

```
                                    <CATEGORY>IDENTICAL</CATEGORY>
                              </COMPU-METHOD>
                              <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"
xsi:type="MIN-MAX-LENGTH-TYPE" TERMINATION="ZERO">
                                    <MAX-LENGTH>256</MAX-LENGTH>
                                    <MIN-LENGTH>1</MIN-LENGTH>
                              </DIAG-CODED-TYPE>
                              <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                        </DATA-OBJECT-PROP>
                  </DATA-OBJECT-PROPS>
            </COMPARAM-SUBSET>
</ODX>
```

The CP_UniqueRespIdTable complex communication parameter is defined as specified in the D-PDU API specification with the additional CP_ECULayerShortName communication parameter added at its end.

```
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
      <COMPARAM-SPEC ID="CP_13391">
            <SHORT-NAME>CP_UDS_on_CAN</SHORT-NAME>
            <PROT-STACKS>
                  <PROT-STACK ID="PS_13392">
                        <SHORT-NAME>UDS_on_CAN</SHORT-NAME>
                        <PDU-PROTOCOL-TYPE>ISO_15765_3_on_ISO_15765_2</PDU-
PROTOCOL-TYPE>
                        <COMPARAM-SUBSET-REFS>
                              <COMPARAM-SUBSET-REF ID-REF="CP_SUBS_11361"
DOCTYPE="COMPARAM-SUBSET" DOCREF="CPS_UDS_on_CAN"/>
                        </COMPARAM-SUBSET-REFS>
                  </PROT-STACK>
            </PROT-STACKS>
      </COMPARAM-SPEC>
</ODX>
```

Next, the functional group and the BASE-VARIANTs are defined (in this case within one DIAG-LAYER-CONTAINER).

```
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=" odx.xsd" MODEL-VERSION="2.1.0">
      <DIAG-LAYER-CONTAINER ID="DLC_14121">
            <SHORT-NAME>DL_Container</SHORT-NAME>
            <FUNCTIONAL-GROUPS>
                  <FUNCTIONAL-GROUP ID="FG_14122">
                        <SHORT-NAME>FG</SHORT-NAME>
                        <DIAG-COMMS>
                              <DIAG-SERVICE ID="DS_14401"
ADDRESSING="FUNCTIONAL">
                                    <SHORT-NAME>FUNC_SAMPLE</SHORT-NAME>
                                    <REQUEST-REF ID-REF="REQ_14412"/>
                                    <POS-RESPONSE-REFS>
                                          <POS-RESPONSE-REF ID-
REF="PR_14581"/>
                                          <POS-RESPONSE-REF ID-
REF="PR_15001"/>
                                    </POS-RESPONSE-REFS>
                                    <NEG-RESPONSE-REFS>
                                          <NEG-RESPONSE-REF ID-
REF="NR_15031"/>
                                    </NEG-RESPONSE-REFS>
                              </DIAG-SERVICE>
                        </DIAG-COMMS>
                        <REQUESTS>
                              <REQUEST ID="REQ_14412">
                                    <SHORT-NAME>FUNC_REQ</SHORT-NAME>
                                    <PARAMS>
```

```
                                                <PARAM xsi:type="CODED-CONST">
                                                      <SHORT-NAME>SID</SHORT-NAME>
                                                      <CODED-VALUE>34</CODED-
VALUE>
                                                      <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                            <BIT-LENGTH>8</BIT-
LENGTH>
                                                      </DIAG-CODED-TYPE>
                                                </PARAM>
                                          </PARAMS>
                                    </REQUEST>
                              </REQUESTS>
                              <POS-RESPONSES>
                                    <POS-RESPONSE ID="PR_14581">
                                          <SHORT-NAME>PR_A</SHORT-NAME>
                                          <PARAMS>
                                                <PARAM xsi:type="CODED-CONST">
                                                      <SHORT-NAME>SID</SHORT-NAME>
                                                      <BYTE-POSITION>0</BYTE-
POSITION>
                                                      <BIT-POSITION>0</BIT-
POSITION>
                                                      <CODED-VALUE>98</CODED-
VALUE>
                                                      <DIAG-CODED-TYPE
xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32">
                                                            <BIT-LENGTH>8</BIT-
LENGTH>
                                                      </DIAG-CODED-TYPE>
                                                </PARAM>
                                          </PARAMS>
                                    </POS-RESPONSE>
                                    <POS-RESPONSE ID="PR_15001">
                                          <SHORT-NAME>PR_B</SHORT-NAME>
                                          <PARAMS>
                                                <PARAM xsi:type="CODED-CONST">
                                                      <SHORT-NAME>SID</SHORT-NAME>
                                                      <BYTE-POSITION>0</BYTE-
POSITION>
                                                      <BIT-POSITION>0</BIT-
POSITION>
                                                      <CODED-VALUE>98</CODED-
VALUE>
                                                      <DIAG-CODED-TYPE
xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32">
                                                            <BIT-LENGTH>8</BIT-
LENGTH>
                                                      </DIAG-CODED-TYPE>
                                                </PARAM>
                                                <PARAM xsi:type="CODED-CONST">
                                                      <SHORT-NAME>Value</SHORT-
NAME>
                                                      <BYTE-POSITION>1</BYTE-
POSITION>
                                                      <BIT-POSITION>0</BIT-
POSITION>
                                                      <CODED-VALUE>312</CODED-
VALUE>
                                                      <DIAG-CODED-TYPE
xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32">
                                                            <BIT-LENGTH>16</BIT-
LENGTH>
                                                      </DIAG-CODED-TYPE>
                                                </PARAM>
                                          </PARAMS>
                                    </POS-RESPONSE>
```

```
                        </POS-RESPONSES>
                        <NEG-RESPONSES>
                              <NEG-RESPONSE ID="NR_15031">
                                    <SHORT-NAME>NR_A</SHORT-NAME>
                                    <PARAMS>
                                          <PARAM xsi:type="CODED-CONST">
                                                <SHORT-NAME>SID</SHORT-NAME>
                                                <BYTE-POSITION>0</BYTE-
POSITION>
                                                <BIT-POSITION>0</BIT-
POSITION>
                                                <CODED-VALUE>127</CODED-
VALUE>
                                                <DIAG-CODED-TYPE
xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32">
                                                      <BIT-LENGTH>8</BIT-
LENGTH>
                                                </DIAG-CODED-TYPE>
                                          </PARAM>
                                          <PARAM xsi:type="VALUE">
                                                <SHORT-NAME>NRC</SHORT-NAME>
                                                <BYTE-POSITION>1</BYTE-
POSITION>
                                                <BIT-POSITION>0</BIT-
POSITION>
                                                <DOP-SNREF SHORT-
NAME="One_byte_identical"/>
                                          </PARAM>
                                    </PARAMS>
                              </NEG-RESPONSE>
                        </NEG-RESPONSES>
                        <COMPARAM-REFS>
                              <COMPARAM-REF ID-REF="CP_14291">
                                    <SIMPLE-VALUE>2001</SIMPLE-VALUE>
                              </COMPARAM-REF>
                              <COMPARAM-REF ID-REF="CCP_11381">
                                    <COMPLEX-VALUE>
                                          <SIMPLE-VALUE>4</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>2018</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>4</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>1970</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>1458</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>A or B</SIMPLE-VALUE>
                                    </COMPLEX-VALUE>
                                    <PROT-STACK-SNREF SHORT-
NAME="UDS_on_CAN"/>
                              </COMPARAM-REF>
                              <COMPARAM-REF ID-REF="CCP_11381">
                                    <COMPLEX-VALUE>
                                          <SIMPLE-VALUE>4</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>2020</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>4</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>1972</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>1460</SIMPLE-VALUE>
                                          <SIMPLE-VALUE>C</SIMPLE-VALUE>
                                    </COMPLEX-VALUE>
                                    <PROT-STACK-SNREF SHORT-
NAME="UDS_on_CAN"/>
                              </COMPARAM-REF>
```

```
                        </COMPARAM-REFS>
                    </FUNCTIONAL-GROUP>
                </FUNCTIONAL-GROUPS>
                <BASE-VARIANTS>
                    <BASE-VARIANT ID="BV_15301">
                        <SHORT-NAME>A</SHORT-NAME>
                        <DIAG-COMMS>
                            <DIAG-SERVICE ID="DS_16011"
ADDRESSING="FUNCTIONAL">
                                <SHORT-NAME>FUNC_SAMPLE</SHORT-NAME>
                                <POS-RESPONSE-REFS>
                                    <POS-RESPONSE-REF ID-
REF="PS_15361"/>
                                </POS-RESPONSE-REFS>
                                <NEG-RESPONSE-REFS>
                                    <NEG-RESPONSE-REF ID-
REF="NR_15381"/>
                                </NEG-RESPONSE-REFS>
                            </DIAG-SERVICE>
                        </DIAG-COMMS>
                        <POS-RESPONSES>
                            <POS-RESPONSE ID="PS_15361">
                                <SHORT-NAME>PR_A</SHORT-NAME>
                                <PARAMS>
                                    <PARAM xsi:type="CODED-CONST">
                                        <SHORT-NAME>SID</SHORT-NAME>
                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                        <BIT-POSITION>0</BIT-
POSITION>
                                        <CODED-VALUE>99</CODED-
VALUE>
                                        <DIAG-CODED-TYPE
xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32">
                                            <BIT-LENGTH>8</BIT-
LENGTH>
                                        </DIAG-CODED-TYPE>
                                    </PARAM>
                                </PARAMS>
                            </POS-RESPONSE>
                        </POS-RESPONSES>
                        <NEG-RESPONSES>
                            <NEG-RESPONSE ID="NR_15381">
                                <SHORT-NAME>NR_A</SHORT-NAME>
                                <PARAMS>
                                    <PARAM xsi:type="CODED-CONST">
                                        <SHORT-NAME>SID</SHORT-NAME>
                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                        <BIT-POSITION>0</BIT-
POSITION>
                                        <CODED-VALUE>127</CODED-
VALUE>
                                        <DIAG-CODED-TYPE
xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32">
                                            <BIT-LENGTH>8</BIT-
LENGTH>
                                        </DIAG-CODED-TYPE>
                                    </PARAM>
                                </PARAMS>
                            </NEG-RESPONSE>
                        </NEG-RESPONSES>
                        <COMPARAM-REFS>
                            <COMPARAM-REF ID-REF="CCP_11381">
                                <COMPLEX-VALUE>
                                    <SIMPLE-VALUE>4</SIMPLE-VALUE>
                                    <SIMPLE-VALUE>2018</SIMPLE-VALUE>
```

```
                                                <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                                <SIMPLE-VALUE>4</SIMPLE-VALUE>
                                                <SIMPLE-VALUE>1970</SIMPLE-VALUE>
                                                <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                                <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                                <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                                <SIMPLE-VALUE>1463</SIMPLE-VALUE>
                                                <SIMPLE-VALUE>A</SIMPLE-VALUE>
                                          </COMPLEX-VALUE>
                                          <PROT-STACK-SNREF SHORT-
NAME="UDS_on_CAN"/>
                                    </COMPARAM-REF>
                              </COMPARAM-REFS>
                        </BASE-VARIANT>
                        <BASE-VARIANT ID="BV_15481">
                              <SHORT-NAME>B</SHORT-NAME>
                        </BASE-VARIANT>
                        <BASE-VARIANT ID="BV_16041">
                              <SHORT-NAME>C</SHORT-NAME>
                              <DIAG-COMMS>
                                    <DIAG-SERVICE ID="DS_16061"
ADDRESSING="FUNCTIONAL">
                                          <SHORT-NAME>FUNC_SAMPLE</SHORT-NAME>
                                          <POS-RESPONSE-REFS>
                                                <POS-RESPONSE-REF ID-
REF="PS_16051"/>
                                          </POS-RESPONSE-REFS>
                                          <NEG-RESPONSE-REFS>
                                                <NEG-RESPONSE-REF ID-
REF="NR_16052"/>
                                          </NEG-RESPONSE-REFS>
                                    </DIAG-SERVICE>
                              </DIAG-COMMS>
                              <POS-RESPONSES>
                                    <POS-RESPONSE ID="PS_16051">
                                          <SHORT-NAME>PR_B</SHORT-NAME>
                                          <PARAMS>
                                                <PARAM xsi:type="CODED-CONST">
                                                      <SHORT-NAME>SID</SHORT-NAME>
                                                      <BYTE-POSITION>0</BYTE-
POSITION>
                                                      <BIT-POSITION>0</BIT-
POSITION>
                                                      <CODED-VALUE>98</CODED-
VALUE>
                                                      <DIAG-CODED-TYPE
xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32">
                                                            <BIT-LENGTH>8</BIT-
LENGTH>
                                                      </DIAG-CODED-TYPE>
                                                </PARAM>
                                          </PARAMS>
                                    </POS-RESPONSE>
                              </POS-RESPONSES>
                              <NEG-RESPONSES>
                                    <NEG-RESPONSE ID="NR_16052">
                                          <SHORT-NAME>NR_A</SHORT-NAME>
                                          <PARAMS>
                                                <PARAM xsi:type="CODED-CONST">
                                                      <SHORT-NAME>SID</SHORT-NAME>
                                                      <BYTE-POSITION>0</BYTE-
POSITION>
                                                      <BIT-POSITION>0</BIT-
POSITION>
                                                      <CODED-VALUE>127</CODED-
VALUE>
```

```
                                                        <DIAG-CODED-TYPE
xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                            </PARAMS>
                                      </NEG-RESPONSE>
                               </NEG-RESPONSES>
                               <COMPARAM-REFS>
                                      <COMPARAM-REF ID-REF="CCP_11381">
                                            <COMPLEX-VALUE>
                                                  <SIMPLE-VALUE>4</SIMPLE-VALUE>
                                                  <SIMPLE-VALUE>2018</SIMPLE-VALUE>
                                                  <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                                  <SIMPLE-VALUE>4</SIMPLE-VALUE>
                                                  <SIMPLE-VALUE>1970</SIMPLE-VALUE>
                                                  <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                                  <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                                  <SIMPLE-VALUE>0</SIMPLE-VALUE>
                                                  <SIMPLE-VALUE>1463</SIMPLE-VALUE>
                                                  <SIMPLE-VALUE>A</SIMPLE-VALUE>
                                            </COMPLEX-VALUE>
                                            <PROT-STACK-SNREF SHORT-
NAME="UDS_on_CAN"/>
                                      </COMPARAM-REF>
                               </COMPARAM-REFS>
                        </BASE-VARIANT>
                  </BASE-VARIANTS>
            </DIAG-LAYER-CONTAINER>
</ODX>
```

7.5 ODX DATA MODEL FOR ECU MEMORY PROGRAMMING

7.5.1 OVERVIEW

One use case of data transfer between any ECU and any runtime system is the programming / (partial) reprogramming of ECU's, called "ECU memory programming". ODX offers a possibility to describe the data needed for an upload or a download procedure. In the following the notation "flash" is also used instead of download. Download in ODX case means data stream from runtime system into ECU, upload means data stream form ECU to runtime system (as opposed to ISO 14230). A diagnostic application can use this information to initiate such a flash process. For this purpose it offers the user a list of all available sessions. After the user has taken his choice, the selected session can be flashed. All needed information, i.e. flash job, compatibility of data with ECU or files containing the actual flash data can be defined within an ODX instance. Even the flash data itself can be defined there. Please refer the ASAM MCD 3 specification for details of flash jobs and their handling.
This chapter specifies the object classes used for programming/reprogramming of ECU's.

7.5.2 ECU-MEM

7.5.2.1 STRUCTURE OF ECU-MEM

Drawing: EcuMem
Package: Flash

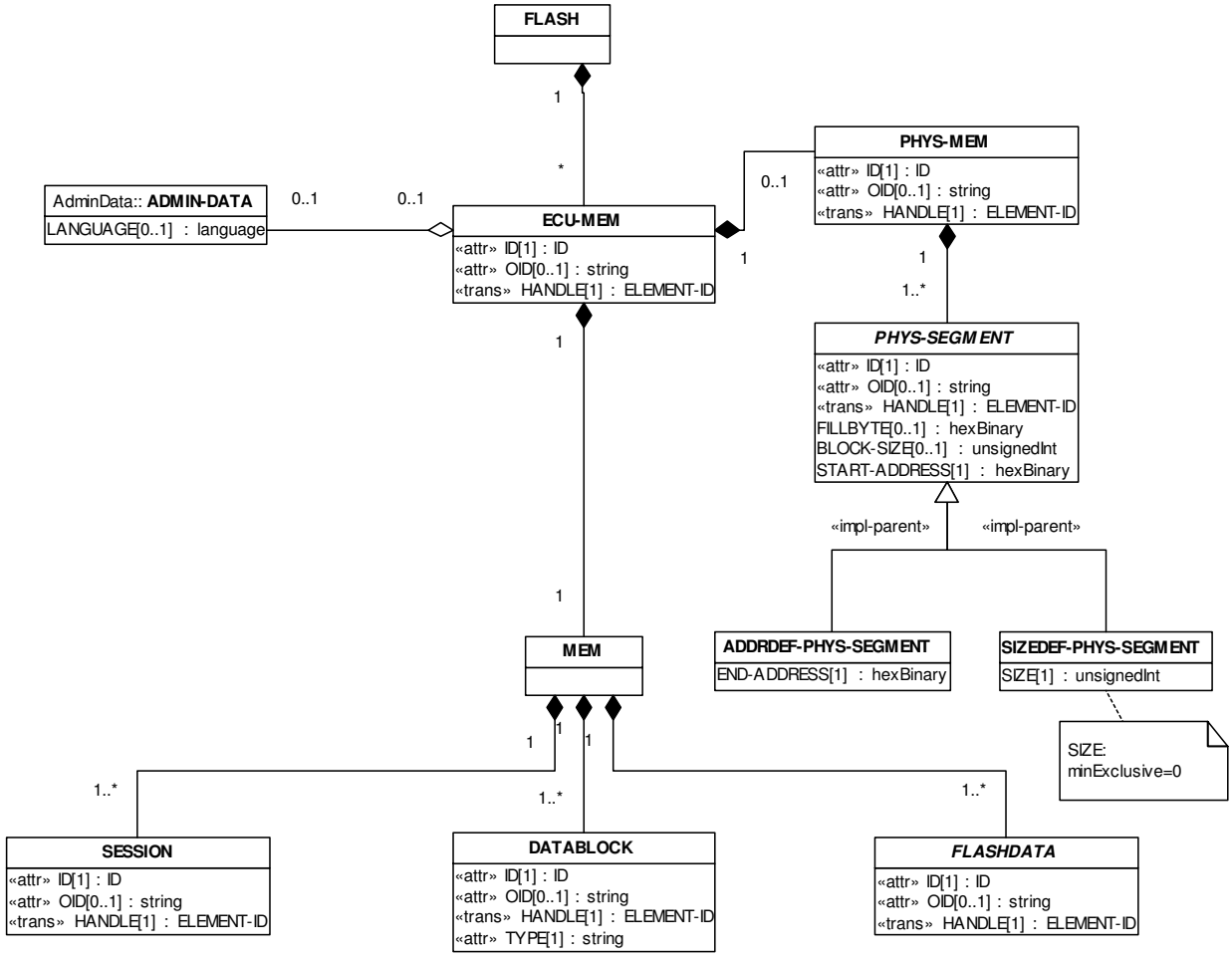


Figure 127 — Structure of ECU-MEM

A single ECU-MEM object is a container for flash data transfer between development, factory and customer. It includes the description of memory of one or more ECUs, given by SESSIONs, DATABLOCKs, FLASHDATAs and PHYS-MEMs.
Typically, an ECU-MEM instance can only be used in conjunction with the appropriate single job (the flash job). For example, it is possible to define manufacturer-specific information like checksum algorithm or encryption method. For this reason an ECU-MEM instance and the flash job that understand the data in this instance should be exchanged together.

7.5.2.2 SESSIONS

SESSIONs are the only logical units that can be chosen for programming. The choice of a session occurs by SESSION-DESC, which references the appropriate SESSION object and the flash job responsible for programming.

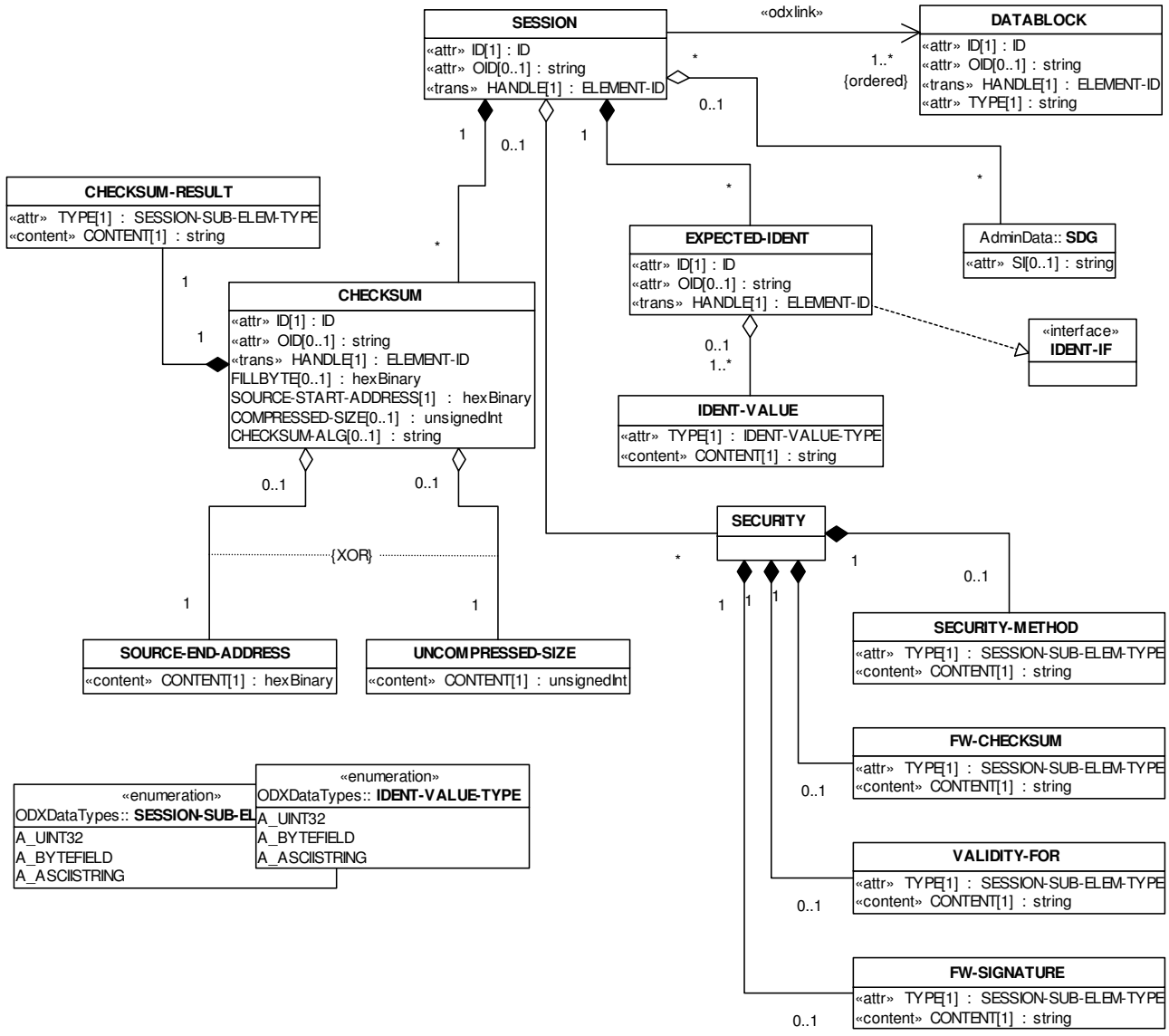


Figure 128 — Structure of SESSION

A SESSION object involves the standard members SHORT-NAME, LONG-NAME, and DESC. It describes which DATABLOCKs are programmed within this session. The DATABLOCKs are referenced by odxlink and are programmed according to the order of the references.
In the following figure an example of an ECU-MEM object is given. There are three sessions in this ECU-MEM. Session1 consists for example of three datablocks:

FlashDriver, Code and Data1. Depending on the selected session different datablocks will be programmed.

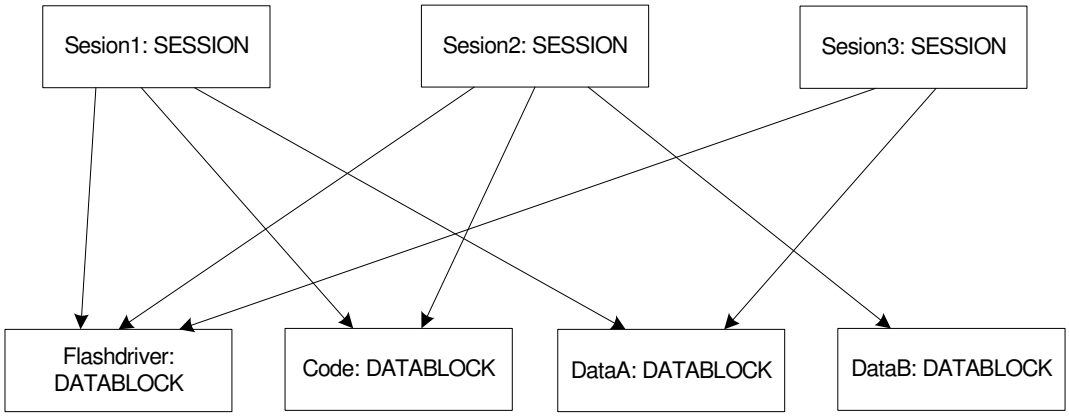


Figure 129 — Example: ECU-MEM with three sessions

EXPECTED-IDENTs hold information regarding the target device. This information can be used to verify if the SESSION is compatible to the target device. This means that the current SESSION can only be programmed into an ECU if all EXPECTED-IDENTs match with the respective idents read from the ECU. To read EXPECTED-IDENTS from ECU the flash job should use a standard protocol command. Each ECU has its own EXPEXTED-IDENTS. For details see also chapter "ECU-MEM Connector".

```
<EXPECTED-IDENTS>
	<EXPECTED-IDENT ID="F.ECUMEM.MEM.SESSION.EXPIDENT01.ID">
		<SHORT-NAME>HARDWARE</SHORT-NAME>
		<LONG-NAME>Hardware</LONG-NAME>
		<DESC>
			<p>Allowed hardware version for this software version</p>
		</DESC>
		<IDENT-VALUES>
			<IDENT-VALUE TYPE="A_ASCIISTRING">hw-value 1</IDENT-VALUE>
			<IDENT-VALUE TYPE="A_ASCIISTRING">hw-value 2</IDENT-VALUE>
		</IDENT-VALUES>
	</EXPECTED-IDENT>
	<EXPECTED-IDENT ID="F.ECUMEM.MEM.SESSION.EXPIDENT02.ID">
		<SHORT-NAME>SOFTWARE2</SHORT-NAME>
		<LONG-NAME>Software 2</LONG-NAME>
		<DESC>
			<p>Allowed software version 2 for this software version</p>
		</DESC>
		<IDENT-VALUES>
			<IDENT-VALUE TYPE="A_ASCIISTRING">sw-value 1</IDENT-VALUE>
			<IDENT-VALUE TYPE="A_ASCIISTRING">sw-value 2</IDENT-VALUE>
		</IDENT-VALUES>
	</EXPECTED-IDENT>
</EXPECTED-IDENTS>
```

Within one session one or more CHECKSUMs may be declared to allow defining of various sets of checksum-result, address range, and algorithm. A CHECKSUM is used to perform checks within the flash process. This information can be used by the flash job to calculate the checksum of the flashed data. If checksum monitoring is performed, the respective algorithm is identified by the name in CHECKSUM-ALG, which may be needed by a job to calculate the CHECKSUM-RESULT. The address range, which is considered for the checksum calculation, is specified by SOURCE-START-ADDR and SOURCE-END-ADDR (or UNCOMPRESSED-SIZE). SOURCE-START-ADDR points to the first byte to be programmed or included in the checksum calculation. SOURCE-END-ADDR

denotes the last byte to be programmed or included in the checksum calculation. Alternatively, UNCOMPRESSED-SIZE can be used to calculate the checksum. COMPRESSED-SIZE holds the size of the compressed SEGMENT/CHECKSUM. The expected result can be found in CHECKSUM-RESULT. A FILLBYTE is used to fill gaps among the logical segments (SEGMENT) in the flash data for checksum calculation of the given address range. SOURCE-START-ADDR denotes the start of an address range in the source data. If the target address inside the ECU differs from the source address of the source data, TARGET-ADDR-OFFSET is used to describe the offset. The SOURCE-START-ADDR is the first address to be included in a checksum calculation. The end of the address range (in target device = ECU) is defined by SOURCE-END-ADDR. This is the last address to be programmed or included in checksum calculation. Alternatively, UNCOMPRESSED-SIZE can be used to calculate the checksum. COMPRESSED-SIZE holds the size of the compressed SEGMENT/CHECKSUM.
With SECURITY it is possible to define security specific information (e.g.: checksum, signature) of the whole session. This information can be used by the flash job to check the integrity and the authenticity of the flash data. The member SECURITY-METHOD denotes the chosen security concept. This might be either single information about the used method or a reference to an external access table. FW-SIGNATURE holds the signature, which has to be sent to the target device (ECU) for verification of authenticity. FW-CHECKSUM holds the checksum (e.g. a CRC32), which has to be sent to the target device for verification of integrity. VALIDITY-FOR describes, which ECU belongs to the given signature. Together with SECURITY-METHOD it is used to influence the behaviour of the flash job.

```
<SECURITYS>
	<SECURITY>
		<SECURITY-METHOD TYPE="A_ASCIISTRING">method_1</SECURITY-METHOD>
		<FW-SIGNATURE TYPE="A_ASCIISTRING">signature_1</FW-SIGNATURE>
		<FW-CHECKSUM TYPE="A_ASCIISTRING">checksum_1</FW-CHECKSUM>
		<VALIDITY-FOR TYPE="A_ASCIISTRING">hw1</VALIDITY-FOR>
	</SECURITY>
</SECURITYS>
```

It is recommended to use SECURITY to check the integrity and authenticity of the source data and CHECKSUM to calculate the checksum of the data flashed to (or stored in) the ECU. It is also reasonable to use SECURITY directly in DATABLOCKs, because they define the actual flash data.

7.5.2.3 DATABLOCKS

A SESSION references one or more DATABLOCKs, which describe the logical structure of the referenced FLASHDATA. A DATABLOCK can describe e.g. a whole code section, a single element like one calibration value; or a flash driver, which has to be loaded into the RAM before the reprogramming routine can be executed.

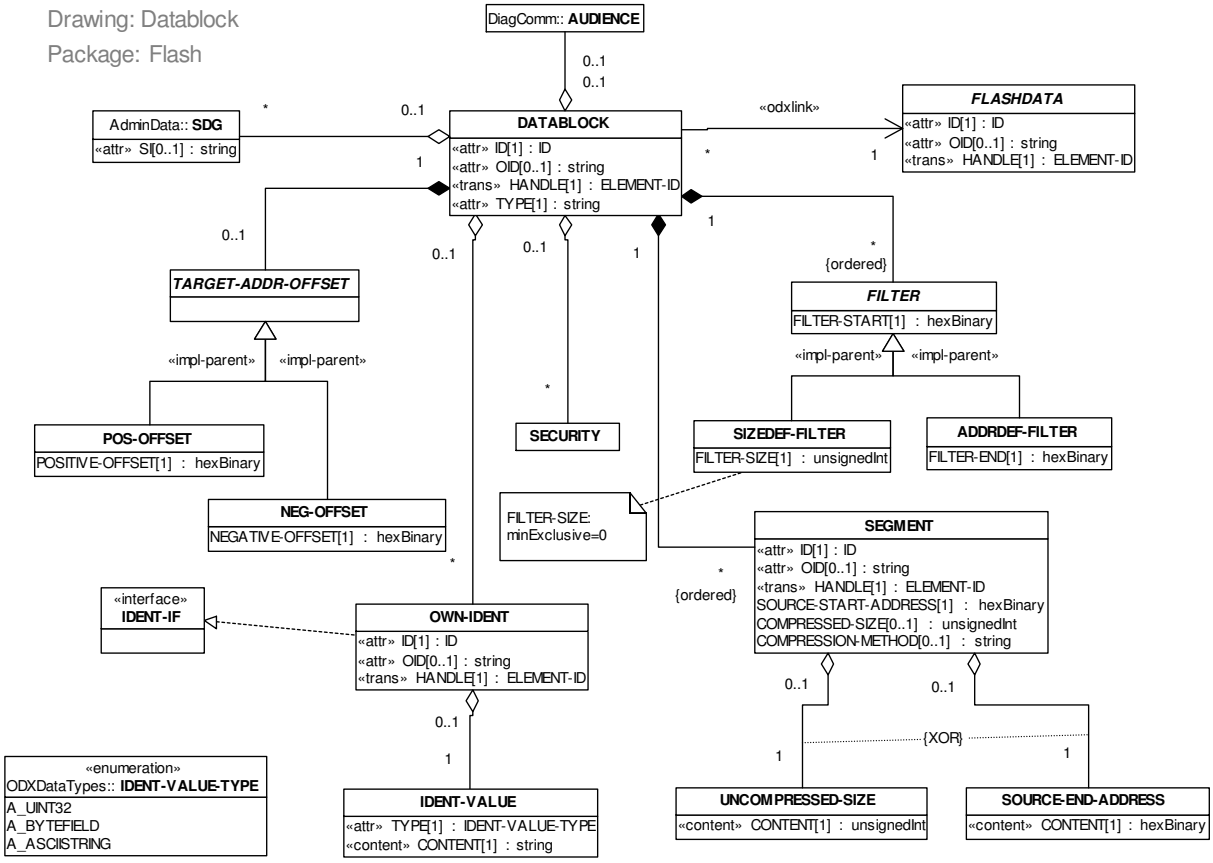


Figure 130 — Structure of datablock

The member TYPE describes the kind of datablock (e.g. "BOOT", "CODE", "DATA", or other values, this TYPE is just a classification and any other proprietary TYPE could be used). This value will be used by a generic programming job to select the method used to unlock or to program the flashdata in this datablock. A flash job can also decide the sequence of programming of these parts.
Like described in the previous section, SECURITY can be used to define security specific information (e.g.: checksum, signature). In this case it is applied to a certain DATABLOCK instead of the whole SESSION.
Each DATABLOCK references to the corresponding FLASHDATA, that contains the actual data described in chapter 7.5.2.4.
FILTERs can be used to reduce the flash data from the source. The data inside the source, which are not covered by one of the filters, will be ignored. Depending on the type of data, a FILTER describes:

— the address range in case of hex formatted data (INTEL-HEX, MOTOROLA-S). Applying of all FILTERs leads to a sub-set of data with the address information taken over from the source.

— the source offset and the length in case of unformatted data (BINARY), where the source can be an external file or inline code defined by DATA in FLASHDATA (see 7.5.2.4). After applying of all FILTERs in this case, we get a new byte stream without any address information.

FILTERs must be applied in the sequence they are defined in ODX. If no FILTER is defined, the whole file (or the content of DATA) will be processed further. Otherwise, the data inside the source is reduced to that part of the data, which fall into one FILTER range (positive filter, pass filter). Address ranges of two different FILTERs must not overlap. The FILTER size must always be greater than zero.
The following filter defines an address range going from 0000h to 1FFFh. The length of this interval equals 2000h:

```
<FILTERS>
       <FILTER xsi:type="ADDRDEF-FILTER">
              <FILTER-START>0000</FILTER-START>
              <FILTER-END>1FFF</FILTER-END>
       </FILTER>
       <FILTER xsi:type="SIZEDEF-FILTER">
              <FILTER-START>2000</FILTER-START>
              <FILTER-SIZE>8192</FILTER-SIZE>
       </FILTER>
</FILTERS>
```

Each DATABLOCK contains at least one SEGMENT, which is given explicitly (by SEGMENTs) or is computed (by any flash tool e.g.) from the address information in the FLASHDATA source. Segments describe the logical structure of DATABLOCK. A segment represents a continuous data stream (without gaps) with a start and an end address. A SEGMENT object defines therefore a SOURCE-START-ADDR and a SOURCE-END-ADDR (or alternatively UNCOMPRESSED-SIZE). Address ranges of two different SEGMENTs must not overlap.
Again, depending on data type, there are two use cases for SEGMENTs:

ii) In case of hex formatted data (INTEL-HEX, MOTOROLA-S) it can be used as a filter in a second step. Herewith it is possible to select a subset of all segments within the address range of the described FILTERS. It is not required to describe SEGMENTs if all SEGMENTs within the address range of the described FILTERs will be programmed. On the other side, each SEGMENT must reference an existing address interval. If a certain address interval does not exist, e.g. because it did not come through the FILTERs, the runtime system should report an error message. The address information of the segments is given by the hex formatted data (file or inline code). Before a SEGMENT can be flashed, the TARGET-ADDR-OFFSET defined in DATABLOCK must be added to all address values.

jj) In case of unformatted data (BINARY) the address information of the SEGMENT shall be used as the target address within the target device (ECU), because the unformatted data contains no further address information. The first SEGMENT starts at the byte position 0 in the FLASHDATA and has the length UNCOMPRESSED-SIZE. If UNCOMPRESSED-SIZE is not given, it is calculated from the SOURCE-START-ADDR and the SOURCE-END-ADDR. The next SEGMENT starts always at the next byte after the last byte of the previous SEGMENT and has the length calculated in the same manner as the length of the first one. After applying of all SEGMENTs, the TARGET-ADDR-OFFSET must be applied in the same way as in the case above. If no SEGMENT is described the reprogramming device has to use the following address information:

1) FILTER-START as SOURCE-START-ADDR and FILTER-END as SOURCE-END-ADDR, if one or more FILTERs are defined

2) 00h as start address and the file length as UNCOMPRESSED-SIZE if there is no FILTER in the DATABLOCK.

```
<SEGMENTS>
       <SEGMENT ID="F.ECUMEM.MEM.SESSION.DATABLOCK.SEGMENT01.ID">
              <SHORT-NAME>SEGMENT1</SHORT-NAME>
              <LONG-NAME>Segment 1</LONG-NAME>
              <SOURCE-START-ADDRESS>0000</SOURCE-START-ADDRESS>
```

```
            <SOURCE-END-ADDRESS>0F21</SOURCE-END-ADDRESS>
      </SEGMENT>
      <SEGMENT ID="F.ECUMEM.MEM.SESSION.DATABLOCK.SEGMENT02.ID">
            <SHORT-NAME>SEGMENT2</SHORT-NAME>
            <LONG-NAME>Segment 2</LONG-NAME>
            <SOURCE-START-ADDRESS>1000</SOURCE-START-ADDRESS>
            <UNCOMPRESSED-SIZE>3889</UNCOMPRESSED-SIZE>
      </SEGMENT>
</SEGMENTS>
```

Within the object TARGET-ADDR-OFFSET the difference between source address and target address (ECU address) can be edited. This can be either a POSITIVE-OFFSET if target address is higher than the source address, or a NEGATIVE-OFFSET if the target address is lower than the source address. In both cases a positive (hexadecimal) value has to be entered. After all segments are determined, TARGET-ADDR-OFFSET is added to (or subtract from) all start and end addresses of the segments. After that the data can be flashed.
A DATABLOCK may contain a list of OWN-IDENTs, which are used for check purposes. E.g. a test device needs the possibility to compare the current software version within the target device (ECU) with the new software version before reprogramming. The information for this comparison is described within the object OWN-IDENT. A particular OWN-IDENT could be e.g. a part number, a software version, or a supplier.

```
<OWN-IDENTS>
      <OWN-IDENT ID="F.ECUMEM.MEM.SESSION.DATABLOCK.OWNIDENT01.ID">
            <SHORT-NAME>SUPPLIER</SHORT-NAME>
            <LONG-NAME>Supplier</LONG-NAME>
            <IDENT-VALUE TYPE="A_ASCIISTRING">John Doe</IDENT-VALUE>
      </OWN-IDENT>
      <OWN-IDENT ID="F.ECUMEM.MEM.SESSION.DATABLOCK.OWNIDENT02.ID">
            <SHORT-NAME>PARTNUMBER</SHORT-NAME>
            <LONG-NAME>Part Number</LONG-NAME>
            <IDENT-VALUE TYPE="A_BYTEFIELD">1A</IDENT-VALUE>
      </OWN-IDENT>
</OWN-IDENTS>
```

Special data groups (SDGS) are used to store OEM-specific data not covered by the ODX data model in a structured form. They are detailed described in section 7.1.2.1.

7.5.2.4 FLASHDATA

FLASHDATA contains the flashdata as inline code or references a file containing it. In the first case an object DATA and in the second case an object DATAFILE must be defined within FLASHDATA. The data defined inline must be represented in ASCII and match XML (for SGML details please refer ISO 8859-1) character set.

Drawing: Flashdata

Package: Flash

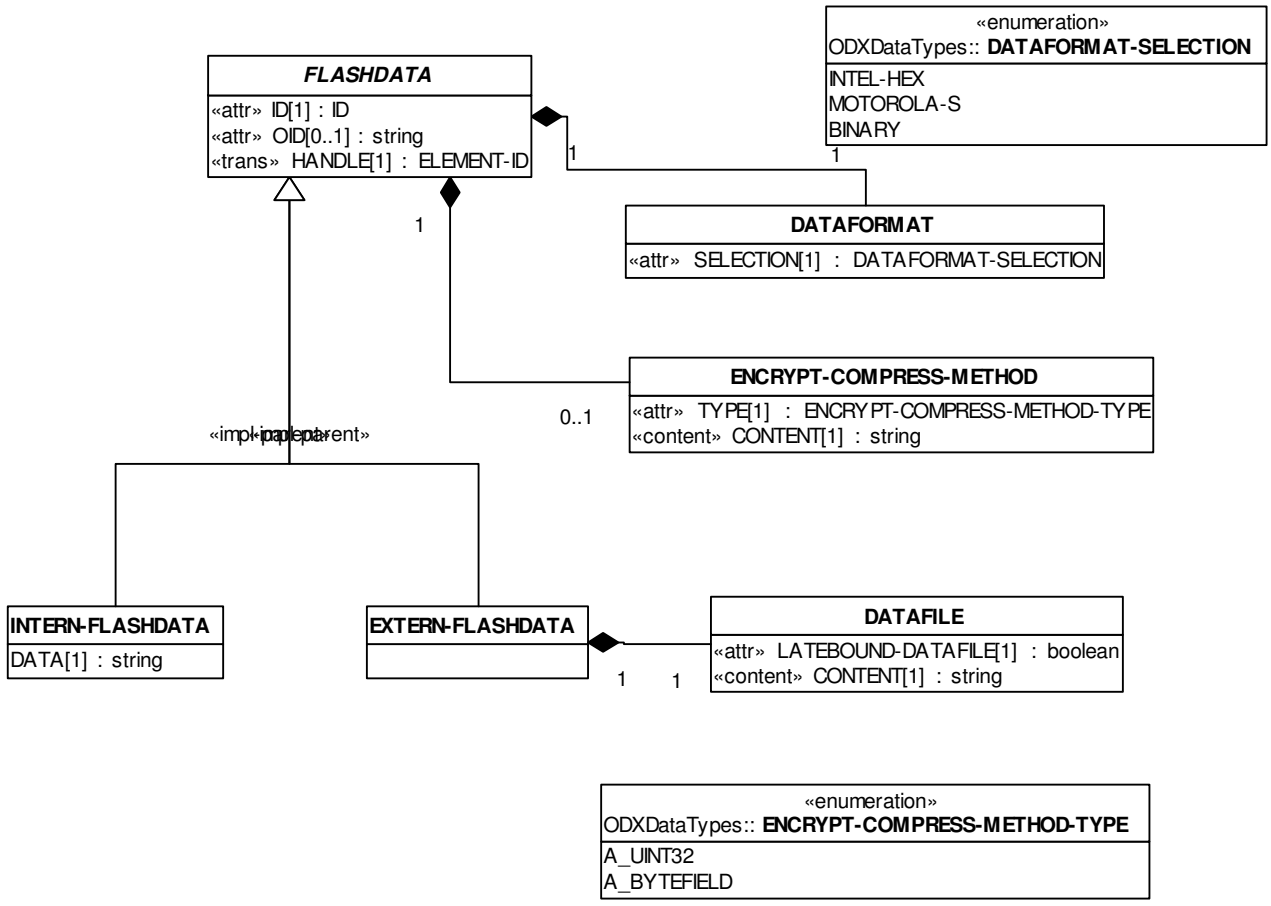


Figure 131 — Structure of flashdata

The member DATAFORMAT specifies the format in which data is represented in the current DATA or the referenced file. The possible values are MOTOROLA-S, INTEL-HEX or BINARY.

EXAMPLE DATA in INTEL-HEX format

```
<FLASHDATA ID="F.ECUMEM.MEM.SESSION.FLASHDATA01.ID" xsi:type="INTERN-FLASHDATA">
       <SHORT-NAME>FLASHDATA01</SHORT-NAME>
       <DATAFORMAT SELECTION="INTEL-HEX"/>
       <DATA>
              :020000020000FC
              :100000003821F64FB80000003800BF193800000052
              :100010003821F64FB80000003821F64FB800000034
              :100020003821F64FB80000003880BC81380000004D
              :100030003821F64FB80000003821F64FB800000014
              :100040003821F64FB8000000770060988000000006B
              :100050003821F64FB80000003821F64FB8000000F4
              ...
       </DATA>
</FLASHDATA> </ECU-MEM>
```

EXAMPLE DATA in MOTOROLA-S format

```
<FLASHDATA ID="F.ECUMEM.MEM.SESSION.FLASHDATA02.ID" xsi:type="INTERN-FLASHDATA">
       <SHORT-NAME>FLASHDATA02</SHORT-NAME>
       <DATAFORMAT SELECTION="MOTOROLA-S"/>
       <DATA>
              S020000020000FC
              S113000003821F64FB80000003800BF193800000052
              S113010003821F64FB80000003821F64FB800000034
              S113020003821F64FB80000003880BC81380000004D
              ...
       </DATA>
</FLASHDATA>
```

If BINARY is used as a type for the file format, the format of the referenced file has to be binary. Each byte of the file can be sent to the ECU without any transformation.
If BINARY is used with DATA, the format of DATA has to be like:

```
<FLASHDATA ID="F.ECUMEM.MEM.SESSION.FLASHDATA03.ID" xsi:type="INTERN-FLASHDATA">
       <SHORT-NAME>FLASHDATA03</SHORT-NAME>
       <DATAFORMAT SELECTION="BINARY"/>
       <DATA>
              110300003821F64FB80000003800BF193800000052
              113010003821F64FB80000003821F64FB800000034
              113020003821F64FB80000003880BC81380000004D
              ...
       </DATA>
</FLASHDATA>
```

Each byte is encoded as 2 hexadecimal digits format (e.g. in UTF8/16: 3F, 00, ...). There is no restriction on the maximum length.
ENCRYPT-COMPRESS-METHOD may describe the parameter "dataFormatIdentifier" of the service "requestDownload" in ISO 14230 (service 34h). To describe how to interpret the values the data type is given by the member TYPE. A detailed description of this data types is given in chapter 7.3.6.2. Here, some examples:

```
<ENCRYPT-COMPRESS-METHOD TYPE="A_UINT32">12</ENCRYPT-COMPRESS-METHOD>
<ENCRYPT-COMPRESS-METHOD TYPE="A_BYTEFIELD">1234</ENCRYPT-COMPRESS-METHOD>
<ENCRYPT-COMPRESS-METHOD TYPE="A_BYTEFIELD">12345678</ENCRYPT-COMPRESS-METHOD>
<ENCRYPT-COMPRESS-METHOD TYPE="A_BYTEFIELD">1A2B3C4D5E6F</ENCRYPT-COMPRESS-
METHOD>
```

In the case (TYPE="A_BYTEFIELD") the length of the parameter results from the length of the specified value.
LATEBOUND-DATAFILE indicates whether the filename is to be determined before the job starts or if the file can be loaded immediately. LATEBOUND-DATAFILE equals “true”

means that the diagnostic application must determine the filename and provide it to the flash job. In case LATEBOUND-DATAFILE “false”, the given filename is used without any interactions with the application. If the filename contains any wildcards ('*' or '?'), the diagnostic application must determine a valid filename before the flash job is started. If there is exactly one file name matching the wildcard the corresponding job is loaded. If there are multiple files matching the wildcard, the application is asked to provide the file name. If the wildcard does not match any file an error is issued by the MCD system. Wildcards can only be used if LATEBOUND-DATAFILE equals “true”.

NOTE In the XML representation all data between the DATA tags is valid and may be used by the flash job including any white spaces (CR, LF, TAB, etc.). Thus, all DATA elements must be whitespace-preserved. But the DATA is hex encoded so it shouldn't be a problem.

7.5.2.5 PHYSICAL-MEMORY

The class PHYS-MEM is used to describe the physical memory layout of an ECU. This memory description may be required by the flash job to check whether logical SEGMENTs do exactly fit to the physical segments defined by PHYS-SEGMENT. For each PHYS-SEGMENT the programming job needs the members START-ADDRESS, END-ADDRESS or SIZE, BLOCKSIZE and FILLBYTE described below. A PHYS-MEM object involves the standard members SHORT-NAME, LONG-NAME, and DESC. DATABLOCKs (described above) are independent from PHYS-MEM.

FILLBYTE describes the byte for filling empty areas to complete the physical segment, i.e. it fills gaps between logical segments in the flash data, if necessary.

BLOCK-SIZE can be used by the flash job to enable parallel programming of memory sub-units to increase the performance of the flash process.

START-ADDRESS describes the first valid address of segment. END-ADDRESS defines the last valid address that belongs to the current segment. SIZE can be used as alternative to END-ADDRESS. It defines the size of the segment in bytes.

EXAMPLE Physical-Memory

```
<PHYS-MEM ID="F.ECUMEM.MEM.PHYSMEM01.ID">
       <SHORT-NAME>PHYSMEM01</SHORT-NAME>
       <LONG-NAME>Physical Memory Map of EXU xyz</LONG-NAME>
       <PHYS-SEGMENTS>
              <PHYS-SEGMENT ID="F.ECUMEM.MEM.PHYSMEM01.PHYSSEG01.ID"
xsi:type="ADDRDEF-PHYS-SEGMENT">
                     <SHORT-NAME>PHYSSEG01</SHORT-NAME>
                     <LONG-NAME>Physical Memory Segment 01</LONG-NAME>
                     <FILLBYTE>FF</FILLBYTE>
                     <BLOCK-SIZE>512</BLOCK-SIZE>
                     <START-ADDRESS>0000</START-ADDRESS>
                     <END-ADDRESS>FFFF</END-ADDRESS>
              </PHYS-SEGMENT>
       </PHYS-SEGMENTS>
</PHYS-MEM>
```

7.5.3 ECU-MEM CONNECTOR

ECU-MEM-CONNECTOR is used to connect the ECU-MEM to the DIAG-LAYER. This part of ODX also links the ECU-MEM-objects with DIAG-COMMS e.g. used to read idents, download or upload the data. It allows an ECU to be flashed using ODX.
The following figure shows the structure of the ECU-MEM-CONNECTOR:

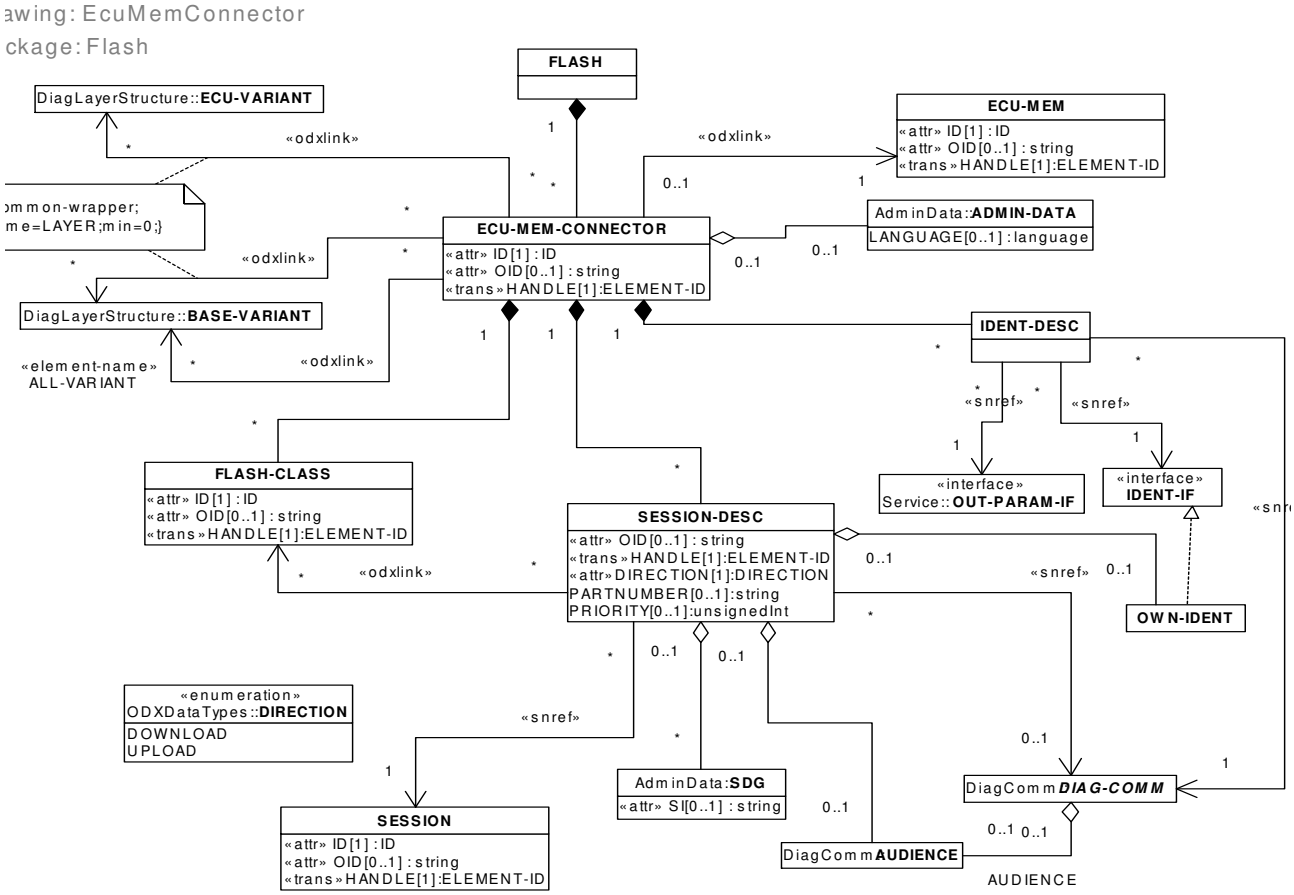


Figure 132 — Structure of ECU-MEM-CONNECTOR

Firstly, an ECU-MEM-CONNECTOR object involves the standard members which are described in chapter 7.1.2.
FLASH-CLASSes can be used to group sessions according to certain criteria. A FLASH-CLASS is defined by SHORT-NAME, LONG-NAME and DESC. In order to reference a class it also have the ID attribute. A link from a SESSION-DESC to a FLASH-CLASS defines the membership of the SESSION-DESC in the referenced class.

SESSION-DESCs are used to store information about sessions and jobs used for download or upload. A particular SESSION-DESC involves the following members: DIRECTION specifies whether it is a description of a download or an upload session. In accordance with that it must be set to "DOWNLAOD" or to "UPLOAD". SHORT-NAME, LONG-NAME, DESC and PARTNUMBER describe the SESSION-DESC object. SHORT-NAME or PARTNUMBER can alternatively be used by tools to allow the selection of concrete sessions for flashing. The member PRIORITY defines the priority of the session. If more than one session is selected for programming, sessions with higher priority will be processed first. The value of PRIORITY may take values greater or equal to 0 (non-negative integers). The highest priority is indicated by the value of 0, the default value is 100. Each SESSION-DESC refers to a SESSION from the ECU-MEM structure via SHORT-NAME. Additionally, a DIAG-COMM object (flash job) in the referenced diagnostic layer is referenced by SESSION-DESC. This DIAG-COMM is responsible for download or upload of the flash data. Special data groups (SDGS) are used to store OEM-specific data not covered by the ODX data model in a structured form. They are described in chapter 7.1.2.1.
IDENT-DESCs includes information about idents (OWN-IDENTs or EXPECTED-IDENTs) and the corresponding services that can read them from the ECU. Idents are used to guarantee the compatibility of the ECU and the data to be flashed. An IDENT-DESC object refers via SHORT-NAME to an object of the interface IDENT-IF (EXPECTED-IDENT or OWN-IDENT) and describes herewith the appropriate ident. The referenced service (or job) and a parameter of this service are used for reading the ident. Both references occur using SHORT-NAME. The short name of the parameter must be unique at the service or job. Each time an EXPECTED-IDENT or an OWN-IDENT is used to check whether the data block may be flashed or not, the short name of the ident is searched in the list of IDENT-DESCs. The names of the service and the response parameter in the found IDENT-DESC object are then used to access the ident stored in the ECU.
. ECU-MEM-CONNECTOR references an ECU-MEM object and one or more BASE-VARIANT or ECU-VARIANT objects. The associated ECU-MEM is valid for all the ECU-VARIANTs and BASE-VARIANTs referenced via LAYER-REF and all the ECU-VARIANTs that inherit from the BASE-VARIANT referenced via ALL-VARIANT-REF excluding the BASE-VARIANT itself. SESSION-DESCs and IDENT-DESCs defined in an ECU-MEM-CONNECTOR may only refer to the SESSIONs and idents of the ECU-MEM referenced by this ECU-MEM-CONNECTOR. The diagnostic layer is referenced via odx-link. It contains services or jobs needed for the entire flash process.
EXAMPLE

```
<ECU-MEM-CONNECTOR ID="F.ECUMEMCON01.ID">
       <SHORT-NAME>Connector1</SHORT-NAME>
       <LONG-NAME>Connector 1</LONG-NAME>
       <FLASH-CLASSS>
              <FLASH-CLASS ID="CLASS-1">
                     <SHORT-NAME>CLASS_1</SHORT-NAME>
              </FLASH-CLASS>
       </FLASH-CLASSS>
       <SESSION-DESCS>
              <SESSION-DESC DIRECTION="DOWNLOAD">
                     <SHORT-NAME>SESSION_1</SHORT-NAME>
                     <PARTNUMBER>1</PARTNUMBER>
                     <PRIORITY>1</PRIORITY>
                     <SESSION-SNREF SHORT-NAME="ECU_MEM_SESSION_1"/>
                     <DIAG-COMM-SNREF SHORT-NAME="FlashService"/>
                     <FLASH-CLASS-REFS>
                            <FLASH-CLASS-REF ID-REF="CLASS-1"/>
                     </FLASH-CLASS-REFS>
```

```
                    <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
             </SESSION-DESC>
       </SESSION-DESCS>
       <IDENT-DESCS>
             <IDENT-DESC>
                    <DIAG-COMM-SNREF SHORT-NAME="ReadIdent"/>
                    <IDENT-IF-SNREF SHORT-NAME="Ident1"/>
                    <OUT-PARAM-IF-SNREF SHORT-NAME="identValue"/>
             </IDENT-DESC>
       </IDENT-DESCS>
       <ECU-MEM-REF ID-REF="F.ECUMEM01.ID" DOCTYPE="FLASH"/>
       <LAYER-REFS>
             <LAYER-REF DOCTYPE="LAYER" DOCREF="ECUVARIANT01" ID-
REF="ECUVARIANT01.ID"/>
       </LAYER-REFS>
</ECU-MEM-CONNECTOR>
```

7.5.4 THE PROGRAMMING PROCESS AS A WHOLE

The whole flash process can be divided into 5 steps:

3) Select a SESSION-DESC in ECU-MEM-CONNECTOR via short-name (or part number)

4) Read session data from ECU-MEM (it is referenced via SHORT-NAME from SESSION-DESC)

5) Check whether flash-ware can be flashed (yes, if all expected idents are present). The data to flash must be compatible with the already existing data or code in the ECU.

6) Load the flash job responsible for flashing in the DIAG-LAYER instance (it is referenced via short-name from SESSION-DESC)

7) Flash every data block, which belongs to the selected session, according to the defined FILTERs and SEGMENTs

In the first step, a SESSION-DESC is selected. It is an entry point into the flash process and is identified by its SHORT-NAME. Additionally, it may have a PARTNUMBER, which must be unique among all SESSION-DESCs in the ODX database. SHORT-NAME and PARTNUMBER can be alternatively used to select a SESSION-DESC, which describes the SESSION to be flashed. In the second step, the appropriate SESSION, referenced via SHORT-NAME from the selected SESSION-DESC, is read.
The step 3) occurs then as follows:

— For all EXPECTED-IDENTs of the selected SESSION, find the corresponding DIAG-COMM in DIAG-LAYER using IDENT-DESC objects in the ECU-MEM-CONNECTOR (see example in Figure 133):

  a. Read the value using the service (or job) and the parameter referenced from IDENT-DESC. The referenced DIAG-COMM is used for reading and the parameter holds the value to be matched.

  b. If at least one IDENT-VALUE defined by the EXPECTED-IDENT object (logical OR) matches the value returned by the DIAG-COMM, the check of this EXPECTED-IDENT is successful.

— If all EXPECTED-IDENTs (logical AND) were successful, the flash data can be flashed.

If the check of EXPECTED-IDENTs was successful, the external specified flash job referenced from the SESSION-DESC is loaded in the next step. This job must be visible in the DIAG-LAYER referenced from ECU-MEM-CONNECTOR, i.e. it must be defined there or inherited from the upper layer.
Finally, each DATABLOCK of the SESSION is flashed in the order of their occurrence in the step 5). The following operations are necessary for that:

- Check whether flash data already exists in the ECU. Therefore, the presence of all OWN-IDENTs is checked in the same way as the presence of EXPECTED-IDENTs was checked in step 3). The presence of all OWN-IDENTs means that the identical data is already exists in the ECU. In this case, the diagnostic system should ask whether it should overwrite the data.

- Apply FILTERs to FLASHDATA according to the description in section 7.5.2.3. If no FILTER is defined in the DATABLOCK, the whole INTERN- or EXTERN-FLASHDATA is used as input for the next operation.

- Apply SEGMENTs to the output of the previous operation according to the description in section 7.5.2.3. If no SEGMENT is defined in the DATABLOCK, calculate segments using the address information defined in FILTERs and in FLASHDATA. It is important, that one segment represents a coherent data area, because only those can be flashed to the ECU.

- Unlock the ECU to flash all segments of the DATABLOCK.

- Write segments to the ECU in the order they are defined in the DATABLOCK or in the order of their start addresses if no SEGMETs are defined (SEGMENTS have been calculated). This order is independent to the order of flash data in any kind of flashware container. Therefore, the flash job, referenced from the SESSION-DESC is used.

As an example, the figure below shows an overview of ODX objects used for flashing. The SESSION-DESC SessDesc1 is selected there, which references the SESSION Sess1 in the ECU-MEM instance. DATABLOCKs referenced by Sess1 should be flashed. In this example there is only one DATABLOCK (Block1), which is flashed if all EXPECTED-IDENTs defined in the session can be found in the ECU. To check this, the IDENT-DESC objects are sought in the ECU-MEM-CONNECTOR, which reference the EXPECTED-IDENTs of the session Sess1. IdentDesc1 references for example the EXPECTED-IDENT Ident1 of this session. It also references the service CheckIdent1Service that is used for reading of a value. The read value is then compared with the IDENT-VALUEs defined in Ident1. In this example, there are two values: 17 and 18. If one of them matches the read value, then the check of Ident1 terminates with success. After that, Ident2 is checked in the same manner. Finally, if both checks were successful, all segments of the DATABLOCK Block1 are calculated. Since there are no SEGMENTs defined in this DATABLOCK, the whole DATA content of Data1 is used as one segment and can be flashed all at once.

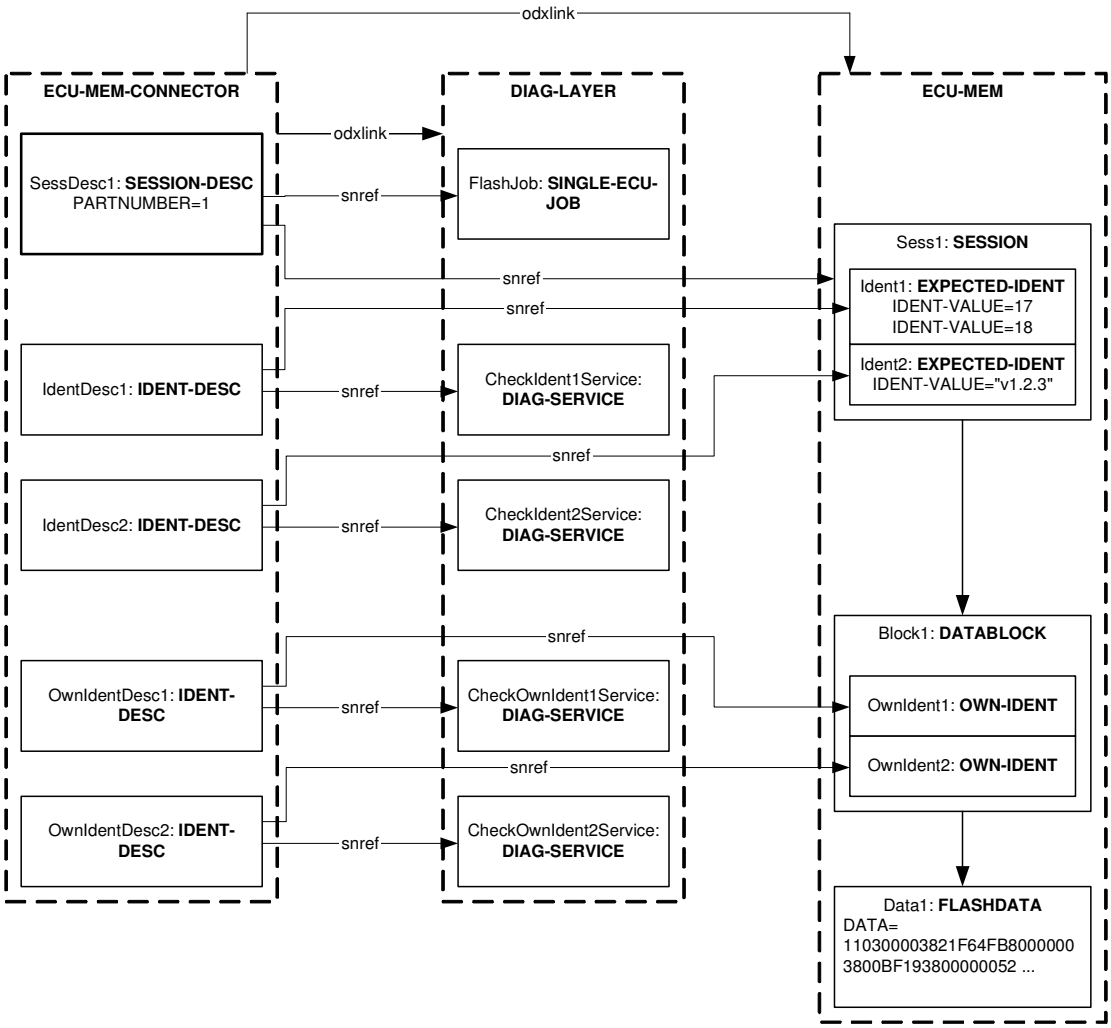


Figure 133 — ECU-MEM-CONNECTOR as mediator between DIAG-LAYER and ECU-MEM

7.5.5 THE UPLOAD PROCESS AS A WHOLE

In contrast to the flash process, during the upload process data are read from the ECU and then are written into a file. The following restrictions are required when an upload is to be processed:

— No DATABLOCK referenced from the selected SESSION may define any FILTER objects

— Every DATABLOCK must reference exactly one file (FLASHDATA with specified DATAFILE). Before uploading starts the file name must be determined.

— At least one SEGMENT object must be defined in each DATABLOCK referenced by the selected SESSION

— The definition of idents (OWN- or EXPECTED-IDENT) is ignored by upload.

The upload process performs in 4 steps:

8) Select a SESSION-DESC in ECU-MEM-CONNECTOR via short-name (or part number)

9) Read information about the appropriate SESSION object from ECU-MEM (it is referenced via SHORT-NAME from SESSION-DESC)

10) Load the job responsible for uploading in the DIAG-LAYER instance (it is referenced via short-name from SESSION-DESC)

11) Read each DATABLOCK, which belongs to the selected session using addresses specified by the SEGMENTs, and store it in the file, specified by this DATABLOCK.

In case the data should be stored in a hex format (INTEL-HEX, MOTOROLA-S), the address information is stored within the destination file. This information complies with the address information given by SEGMENT objects of each DATABLOCK.
In case of BINARY no address information is stored within the destination file. Thus, the received data is stored in the file in the order of DATABLOCKs and SEGMENTs in the ODX instance.

7.6 USAGE SCENARIOS (FLASH)

To understand the handling of the ECU-MEM easier this chapter explains the usage of the important elements by an exemplary ECU.
This ECU model shall dispose of one microcontroller with four separate memories. The first memory area is reserved for the flash driver (8kByte boot code). The second memory area contains the program code and has a size of 120kBytes. The third and the fourth memory area are reserved for data (16kByte DataA and 16kByte DataB). Altogether there are four physically independent memory segments which could be programmed. In this example there are two software versions (V01 and V02) for each memory segment and two different versions of ECU hardware.

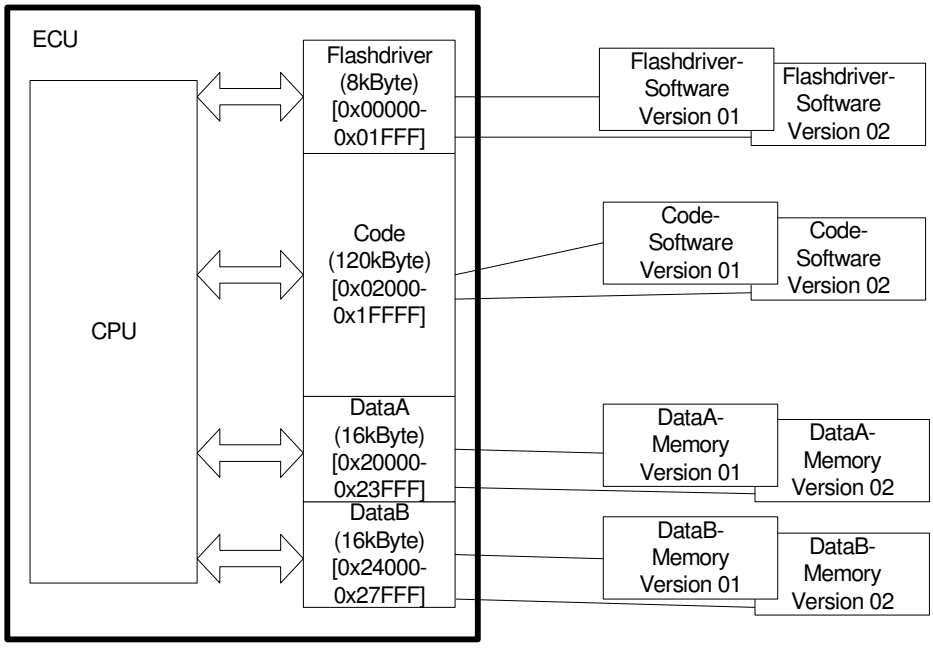


Figure 134 — Hard- and Software of an exemplary ECU

Because different software versions may not be compatible to any hardware and also the content of the data memory depends on the software version of the code, all valid combinations shall be arranged in different sessions. The differentiation, which SESSION could be used with the respective ECU hardware or software is done by the EXPECTED-

IDENT elements. The table bellow shows the dependencies between the ECU hardware and the possible software versions.

Table 13 — Example for valid software combinations

Combination	Hardware	Flashdriver	Code	DataA	DataB
1	Version 1	Version 1	Version 1	Version 1	Version 1
2	Version 1	Version 1	Version 1	Version 1	Version 2
3	Version 2	Version 2	Version 2	Version 1	Version 2
4	Version 2	Version 2	Version 2	Version 2	Version 2

The four possible combinations of hard- and software versions could be mapped to different flash SESSIONS. The elements EXPECTED-IDENT in each SESSION enables the tester to compare the identification values read form the ECU with the expected values of the available SESSIONs. Only the SESSION with matching identification parameters (valid flash sessions for the current ECU) shall be possible to be programmed by the flash tool.
The following ECU-MEM instance describes the FLASH instance for an ECU with the properties mentioned above. The first session is valid for ECUs with hardware version 'V01'. All DATABLOCKs of these SESSION can be programmed if the ECU hardware is matching.
The complete file can be found in Annex B.

```
<FLASH ID="FLASH01.ID">
       <SHORT-NAME>FLASH_ECU</SHORT-NAME>
       <ECU-MEMS>
              <ECU-MEM ID="F.ECUMEM01.ID">
                     <SHORT-NAME>ECUMEM_FLASH_ECU</SHORT-NAME>
                     <MEM>
                            <SESSIONS>
                                   <SESSION ID="F.ECUMEM.MEM.SESSION01.ID">
                                   <SHORT-NAME>FLASH_ECU_HW1_COMPLETE</SHORT-NAME>
                                   <LONG-NAME>Flash ECU with hardware version 'V01'
complete</LONG-NAME>
                                   <EXPECTED-IDENTS>
                                          <EXPECTED-IDENT
ID="F.ECUMEM.MEM.SESSION.EXPIDENT01.ID">
                                                 <SHORT-
NAME>HARDWARE_VERSION</SHORT-NAME>
                                                 <IDENT-VALUES>
                                                        <IDENT-VALUE
TYPE="A_BYTEFIELD">01</IDENT-VALUE>
                                                 </IDENT-VALUES>
                                          </EXPECTED-IDENT>
                                   </EXPECTED-IDENTS>
```

Inside the SECURITYS branch a SIGNATURE for the current SESSION is defined. With the SECURITY-METHOD and the SIGNATURE the program tool or the ECU can be enabled to check the consistency and the authenticity. One possibility is to transmit the signature to the ECU after programming the memory during execution of the flash job. The ECU compares the signature with value calculated internally (e.g. by public key and flash-functions).

```
<SECURITYS>
       <SECURITY>
              <SECURITY-METHOD TYPE="A_ASCIISTRING">SECMETH_SIG01</SECURITY-
METHOD>
              <FW-SIGNATURE TYPE="A_BYTEFIELD">0A5F37</FW-SIGNATURE>
       </SECURITY>
</SECURITYS>
```

The DATABLOCK-REFS contain the references to the DATABLOCKs which belong to this session. In this session four DATABLOCKs are referenced for programming all four memory segments of the ECU.

```
<DATABLOCK-REFS>
      <DATABLOCK-REF ID-REF="F.ECUMEM.MEM.DATABLOCK01.ID" DOCTYPE="FLASH"/>
      <DATABLOCK-REF ID-REF="F.ECUMEM.MEM.DATABLOCK02.ID" DOCTYPE="FLASH"/>
      <DATABLOCK-REF ID-REF="F.ECUMEM.MEM.DATABLOCK03.ID" DOCTYPE="FLASH"/>
      <DATABLOCK-REF ID-REF="F.ECUMEM.MEM.DATABLOCK04.ID" DOCTYPE="FLASH"/>
</DATABLOCK-REFS>
```

The next SESSION is only for updating the memory area reserved for DataB 'V02'. Because the new data version is running on hardware 'V01' only if Code and DataA is 'V01' as well, three EXPECTED-IDENT elements will be needed. A flash tool must process this SESSION only if all identifications read from the ECU do match with the values in the EXPECTED-IDENTs.

```
</SESSION>
      <SESSION ID="F.ECUMEM.MEM.SESSION02.ID">
            <SHORT-NAME>UPDATE_DATAB</SHORT-NAME>
            <LONG-NAME>Update DataB segment to version 'V02'</LONG-NAME>
            <EXPECTED-IDENTS>
                  <EXPECTED-IDENT ID="F.ECUMEM.MEM.SESSION.EXPIDENT02.ID">
                        <SHORT-NAME>HARDWARE_VERSION</SHORT-NAME>
                        <IDENT-VALUES>
                              <IDENT-VALUE TYPE="A_BYTEFIELD">01</IDENT-VALUE>
                        </IDENT-VALUES>
                  </EXPECTED-IDENT>
                  <EXPECTED-IDENT ID="F.ECUMEM.MEM.ESSION.EXPIDENT03.ID">
                        <SHORT-NAME>CODE_VERSION</SHORT-NAME>
                        <IDENT-VALUES>
                              <IDENT-VALUE TYPE="A_BYTEFIELD">01</IDENT-VALUE>
                        </IDENT-VALUES>
                  </EXPECTED-IDENT>
                  <EXPECTED-IDENT ID="F.ECUMEM.MEM.SESSION.EXPIDENT04.ID">
                        <SHORT-NAME>DATAA_VERSION</SHORT-NAME>
                        <IDENT-VALUES>
                              <IDENT-VALUE TYPE="A_BYTEFIELD">01</IDENT-VALUE>
                        </IDENT-VALUES>
                  </EXPECTED-IDENT>
            </EXPECTED-IDENTS>
```

The data memory is not fused by a signature. Only a checksum is defined to make sure the integrity of the programmed data.

```
      <CHECKSUMS>
            <CHECKSUM ID="F.ECUMEM..MEM.SESSION.CHECKSUM01.ID">
                  <SHORT-NAME>CHECKSUM_DATAB_V02</SHORT-NAME>
                  <FILLBYTE>FF</FILLBYTE>
                  <SOURCE-START-ADDRESS>024000</SOURCE-START-ADDRESS>
                  <CHECKSUM-ALG>CHECKALGO_01</CHECKSUM-ALG>
                  <SOURCE-END-ADDRESS>027FFF</SOURCE-END-ADDRESS>
                  <CHECKSUM-RESULT TYPE="A_BYTEFIELD">55B0</CHECKSUM-RESULT>
            </CHECKSUM>
      </CHECKSUMS>
```

Inside DATABLOCK-REFS of this SESSION the DATABLOCK for DataB memory is referenced.

```
      <DATABLOCK-REFS>
            <DATABLOCK-REF ID-REF="F.ECUMEM.MEM.DATABLOCK09.ID"
      DOCTYPE="FLASH"/>
      </DATABLOCK-REFS>
      </SESSION>
```

The third SESSION allows an update of the flash driver and the code memory to software version 'V02'. This update is independent from the hardware version or other software components of the ECU. A flash tool may program this SESSION without compare any identifications read from the ECU.

```
            <SESSION ID="F.ECUMEM.MEM.SESSION03.ID">
                  <SHORT-NAME>UPDATE_BOOT_AND_CODE_HW1</SHORT-NAME>
                  <LONG-NAME>Update BOOT and CODE segments on hardware version
'V01' to software versions 'V02'</LONG-NAME>
```

Each of the flashdriver and code memory segment need also a SECURITY element including the SIGNATURE. In this example the SIGNATURE is attached to the DATABLOCK for the code memory.

```
                  <DATABLOCK-REFS>
                        <DATABLOCK-REF ID-REF="F.ECUMEM.MEM.DATABLOCK05.ID"
DOCTYPE="FLASH"/>
                        <DATABLOCK-REF ID-REF="F.ECUMEM.MEM.DATABLOCK06.ID"
DOCTYPE="FLASH"/>
                  </DATABLOCK-REFS>
            </SESSION>
```

The fourth session updates both data memory areas of an ECU with hardware version 'V02' and Code version 'V02'. The elements in this session are similar to the second session.

```
            <SESSION ID="F.ECUMEM.MEM.SESSION04.ID">
                  <SHORT-NAME>UPDATE_DATA_HW2_COMPLETE</SHORT-NAME>
                  <LONG-NAME>Update DataA and DataB segments on hardware
version 'V02'  to data versions 'V02'</LONG-NAME>
                  <EXPECTED-IDENTS>
                        <EXPECTED-IDENT
ID="F.ECUMEM.MEM.SESSION.EXPIDENT05.ID">
                              <SHORT-NAME>HARDWARE_VERSION</SHORT-NAME>
                              <IDENT-VALUES>
                                    <IDENT-VALUE
TYPE="A_BYTEFIELD">02</IDENT-VALUE>
                              </IDENT-VALUES>
                        </EXPECTED-IDENT>
                        <EXPECTED-IDENT
ID="F.ECUMEM.MEM.SESSION.EXPIDENT06.ID">
                              <SHORT-NAME>CODE_VERSION</SHORT-NAME>
                              <IDENT-VALUES>
                                    <IDENT-VALUE
TYPE="A_BYTEFIELD">02</IDENT-VALUE>
                              </IDENT-VALUES>
                        </EXPECTED-IDENT>
                  </EXPECTED-IDENTS>
                  <CHECKSUMS>
                        <CHECKSUM ID="F.ECUMEM.MEM.SESSION.CHECKSUM02.ID">
                              <SHORT-NAME>CHECKSUM_DATAA_V02</SHORT-NAME>
                              <FILLBYTE>FF</FILLBYTE>
                              <SOURCE-START-ADDRESS>020000</SOURCE-START-
ADDRESS>
                              <CHECKSUM-ALG>CHECKALGO_01</CHECKSUM-ALG>
                              <SOURCE-END-ADDRESS>023FFF</SOURCE-END-ADDRESS>
                              <CHECKSUM-RESULT
TYPE="A_BYTEFIELD">56A9</CHECKSUM-RESULT>
                        </CHECKSUM>
                        <CHECKSUM ID="F.ECUMEM.MEM.SESSION.CHECKSUM03.ID">
                              <SHORT-NAME>CHECKSUM_DATAB_V02</SHORT-NAME>
                              <FILLBYTE>FF</FILLBYTE>
                              <SOURCE-START-ADDRESS>024000</SOURCE-START-
ADDRESS>
                              <CHECKSUM-ALG>CHECKALGO_01</CHECKSUM-ALG>
                              <SOURCE-END-ADDRESS>027FFF</SOURCE-END-ADDRESS>
```

```
                                    <CHECKSUM-RESULT
TYPE="A_BYTEFIELD">55B0</CHECKSUM-RESULT>
                              </CHECKSUM>
                        </CHECKSUMS>
                        <DATABLOCK-REFS>
                              <DATABLOCK-REF ID-REF="F.ECUMEM.MEM.DATABLOCK07.ID"
DOCTYPE="FLASH"/>
                              <DATABLOCK-REF ID-REF="F.ECUMEM.MEM.DATABLOCK08.ID"
DOCTYPE="FLASH"/>
                        </DATABLOCK-REFS>
                  </SESSION>
            </SESSIONS>
```

After the SESSIONS all referenced DATABLOCKs need to be defined inside the DATABLOCKS branch. The first DATABLOCK references the code for the flash driver version 'V01'.

```
            <DATABLOCKS>
                  <DATABLOCK ID="F.ECUMEM.MEM.DATABLOCK01.ID" TYPE="CODE">
                        <SHORT-NAME>BOOT_V01</SHORT-NAME>
                        <FLASHDATA-REF ID-REF="F.ECUMEM.MEM.FLASHDATA01.ID"
DOCTYPE="FLASH"/>
                        <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true" IS-
AFTERSALES="true"
                          IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
                  </DATABLOCK>
```

The second DATABLOCK references the software for the code segment version 'V01'.

```
                  <DATABLOCK ID="F.ECUMEM.MEM.DATABLOCK02.ID" TYPE="CODE">
                        <SHORT-NAME>CODE_V01</SHORT-NAME>
                        <FLASHDATA-REF ID-REF="F.ECUMEM.MEM.FLASHDATA02.ID"
DOCTYPE="FLASH"/>
                        <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true" IS-
AFTERSALES="true"
                          IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
                  </DATABLOCK>
```

The third and the fourth DATABLOCK references the memory for DataA and DataB segments. Because the data-source is in binary format, the element SEGMENT shall be used to inform about the target address. The element TARGET-ADDR-OFFSET is optional and represents the start address offset for programming this block on the target. BINARY code does not contain any other address information in contrast to INTEL-HEX or MOTOROLA-S format.

```
<DATABLOCK ID="F.ECUMEM.MEM.DATABLOCK03.ID" TYPE="DATA">
                        <SHORT-NAME>DATAA_V01</SHORT-NAME>
                        <FLASHDATA-REF ID-REF="F.ECUMEM.MEM.FLASHDATA03.ID"
DOCTYPE="FLASH"/>
                        <SEGMENTS>
                              <SEGMENT ID="F.ECUMEM.MEM.DATABLOCK.SEGMENT01.ID">
                                    <SHORT-NAME>DATAA_SEG_V01</SHORT-NAME>
                                    <SOURCE-START-ADDRESS>00</SOURCE-START-ADDRESS>
                                    <UNCOMPRESSED-SIZE>8192</UNCOMPRESSED-SIZE>
                              </SEGMENT>
                        </SEGMENTS>
                        <TARGET-ADDR-OFFSET xsi:type="POS-OFFSET">
                              <POSITIVE-OFFSET>020000</POSITIVE-OFFSET>
                        </TARGET-ADDR-OFFSET>
                        <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
                  </DATABLOCK>
                  <DATABLOCK ID="F.ECUMEM.MEM.DATABLOCK04.ID" TYPE="DATA">
                        <SHORT-NAME>DATAB_V01</SHORT-NAME>
                        <FLASHDATA-REF ID-REF="F.ECUMEM.MEM.FLASHDATA04.ID"
DOCTYPE="FLASH"/>
                        <SEGMENTS>
```

```
                        <SEGMENT ID="F.ECUMEM.MEM.DATABLOCK.SEGMENT02.ID">
                              <SHORT-NAME>DATAB_SEG_V01</SHORT-NAME>
                              <SOURCE-START-ADDRESS>00</SOURCE-START-ADDRESS>
                              <UNCOMPRESSED-SIZE>8192</UNCOMPRESSED-SIZE>
                        </SEGMENT>
                  </SEGMENTS>
                  <TARGET-ADDR-OFFSET xsi:type="POS-OFFSET">
                        <POSITIVE-OFFSET>024000</POSITIVE-OFFSET>
                  </TARGET-ADDR-OFFSET>
                  <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
            </DATABLOCK>
```

DATABLOCK five has the same structure as DATABLOCK one and references the flash driver code version 'V02'.

```
            <DATABLOCK ID="F.ECUMEM.MEM.DATABLOCK05.ID" TYPE="CODE">
                  <SHORT-NAME>BOOT_V02</SHORT-NAME>
                  <FLASHDATA-REF ID-REF="F.ECUMEM.MEM.FLASHDATA05.ID"
DOCTYPE="FLASH"/>
            </DATABLOCK>
```

The sixth DATABLOCK reference the code memory version 'V02'. Because this DATABLOCK is only used in a SESSION for updating the memory it is not necessary to replace the complete memory segments. With the filters only the different memory areas will masked from the source and transferred to the ECU. In this DATABLOCK only the memory beginning form address 2000h up to 17FFFh need to be changed. An additional security element is assigned to the DATABLOCK, which includes the SIGNATURE for the part of code.

```
            <DATABLOCK ID="F.ECUMEM.MEM.DATABLOCK06.ID" TYPE="CODE">
                  <SHORT-NAME>CODE_V02</SHORT-NAME>
                  <FLASHDATA-REF ID-REF="F.ECUMEM.MEM.FLASHDATA06.ID"
DOCTYPE="FLASH"/>
                  <FILTERS>
                        <FILTER xsi:type="ADDRDEF-FILTER">
                              <FILTER-START>2000</FILTER-START>
                              <FILTER-END>017FFF</FILTER-END>
                        </FILTER>
                  </FILTERS>
                  <SECURITYS>
                        <SECURITY>
                              <SECURITY-METHOD
TYPE="A_ASCIISTRING">SECMETH_SIG01</SECURITY-METHOD>
                              <FW-SIGNATURE TYPE="A_BYTEFIELD">1F62BA</FW-
SIGNATURE>
                        </SECURITY>
                  </SECURITYS>
                  <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
            </DATABLOCK>
```

The next two DATABLOCKs have the same structure like the other DATABLOCKs for DataA and DataB defined before. The difference between these DATABLOCKs is the new version ('V02' instead 'V01').

```
            <DATABLOCK ID="F.ECUMEM.MEM.DATABLOCK07.ID" TYPE="DATA">
                  <SHORT-NAME>DATAA_V02</SHORT-NAME>
                  <FLASHDATA-REF ID-REF="F.ECUMEM.MEM.FLASHDATA07.ID"
DOCTYPE="FLASH"/>
                  <SEGMENTS>
                        <SEGMENT ID="F.ECUMEM.MEM.DATABLOCK.SEGMENT03.ID">
                              <SHORT-NAME>DATAA_SEG_V02</SHORT-NAME>
                              <SOURCE-START-ADDRESS>00</SOURCE-START-ADDRESS>
                              <UNCOMPRESSED-SIZE>8192</UNCOMPRESSED-SIZE>
                        </SEGMENT>
```

```
                    </SEGMENTS>
                    <TARGET-ADDR-OFFSET xsi:type="POS-OFFSET">
                          <POSITIVE-OFFSET>020000</POSITIVE-OFFSET>
                    </TARGET-ADDR-OFFSET>
                    <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
             </DATABLOCK>
             <DATABLOCK ID="F.ECUMEM.MEM.DATABLOCK08.ID" TYPE="DATA">
                    <SHORT-NAME>DATAB_V02</SHORT-NAME>
                    <FLASHDATA-REF ID-REF="F.ECUMEM.MEM.FLASHDATA08.ID"
DOCTYPE="FLASH"/>
                    <SEGMENTS>
                          <SEGMENT ID="F.ECUMEM.MEM.DATABLOCK.SEGMENT04.ID">
                                <SHORT-NAME>DATAB_SEG_V02</SHORT-NAME>
                                <SOURCE-START-ADDRESS>00</SOURCE-START-ADDRESS>
                                <UNCOMPRESSED-SIZE>8192</UNCOMPRESSED-SIZE>
                          </SEGMENT>
                    </SEGMENTS>
                    <TARGET-ADDR-OFFSET xsi:type="POS-OFFSET">
                          <POSITIVE-OFFSET>024000</POSITIVE-OFFSET>
                    </TARGET-ADDR-OFFSET>
             <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
             </DATABLOCK>
```

The last DATABLOCK is used by the second SESSION for updating DataB defined above. Because this SESSION is also an update, it is not be necessary to exchange the contents of the whole memory segment. The element <SOURCE-START-ADDRESS> is used as pointer in the data source and represents the beginning of the data, which need to be changed. The element <POSITIVE-OFFSET> is used as pointer in the target device memory.

```
      <DATABLOCK ID="F.ECUMEM.MEM.DATABLOCK09.ID" TYPE="DATA">
                    <SHORT-NAME>DATAB_V02_PART1</SHORT-NAME>
                    <FLASHDATA-REF ID-REF="F.ECUMEM.MEM.FLASHDATA08.ID"
DOCTYPE="FLASH"/>
                    <SEGMENTS>
                          <SEGMENT ID="F.ECUMEM.MEM.DATABLOCK.SEGMENT05.ID">
                                <SHORT-NAME>DATAB_SEGPART_V02</SHORT-NAME>
                                <SOURCE-START-ADDRESS>02000</SOURCE-START-
ADDRESS>
                                <UNCOMPRESSED-SIZE>4096</UNCOMPRESSED-SIZE>
                          </SEGMENT>
                    </SEGMENTS>
                    <TARGET-ADDR-OFFSET xsi:type="POS-OFFSET">
                          <POSITIVE-OFFSET>026000</POSITIVE-OFFSET>
                    </TARGET-ADDR-OFFSET>                    <AUDIENCE IS-
MANUFACTURING="true"
                          IS-SUPPLIER="true" IS-AFTERSALES="true"
               IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
             </DATABLOCK>
      </DATABLOCKS>
```

After the DATABLOCKS the FLASHDATAS need to be defined. As described before the source format for data memory is BINARY. The code area and the flash driver should be INTEL-HEX format in this example.
The first FLASHDATA block contains the memory for the flash driver - version 'V01':

```
      <FLASHDATAS>
             <FLASHDATA ID="F.ECUMEM.MEM.FLASHDATA01.ID" xsi:type="INTERN-
FLASHDATA">
                    <SHORT-NAME>FLASHDATA_BOOT_V01</SHORT-NAME>
                    <DATAFORMAT SELECTION="INTEL-HEX"/>
                    <DATA>
:020000020000FC
:100000003821F64FB80000003800BF193800000052
:100010003821F64FB80000003821F64FB800000034
```

```
...
                    </DATA>
               </FLASHDATA>
```

The second FLASHDATA block contains the memory for the code - version 'V01':

```
               <FLASHDATA ID="F.ECUMEM.MEM.FLASHDATA02.ID" xsi:type="INTERN-
FLASHDATA">
                    <SHORT-NAME>FLASHDATA_CODE_V01</SHORT-NAME>
                    <DATAFORMAT SELECTION="INTEL-HEX"/>
                    <DATA>
:020000020000FC
:100000003821F64FB80000003800BF193800000052
:100010003821F64FB80000003821F64FB800000034
...
                    </DATA>
               </FLASHDATA>
```

The third block contains the data for DataA in BINARY format - version 'V01'. Because the binary data can be encrypted in the source (i.e. for engine control modules) an additional element ENCRYPT-COMPRESS-METHOD is necessary for the encryption in the flash tool or ECU:

```
               <FLASHDATA ID="F.ECUMEM.MEM.FLASHDATA03.ID" xsi:type="INTERN-
FLASHDATA">
                    <SHORT-NAME>FLASHDATA_DATAA_V01</SHORT-NAME>
                    <DATAFORMAT SELECTION="BINARY"/>
                    <ENCRYPT-COMPRESS-METHOD TYPE="A_BYTEFIELD">4A</ENCRYPT-
COMPRESS-METHOD>
                    <DATA>
110300003821F64FB80000003800BF193800000052
113010003821F64FB80000003821F64FB800000034
113020003821F64FB80000003880BC81380000004D
...
                    </DATA>
               </FLASHDATA>
```

The fourth block looks similar to block three and contains the data for the DataB segment - version 'V01'.

```
               <FLASHDATA ID="F.ECUMEM.MEM.FLASHDATA04.ID" xsi:type="INTERN-
FLASHDATA">
                    <SHORT-NAME>FLASHDATA_DATAB_V01</SHORT-NAME>
                    <DATAFORMAT SELECTION="BINARY"/>
                    <ENCRYPT-COMPRESS-METHOD TYPE="A_BYTEFIELD">4A</ENCRYPT-
COMPRESS-METHOD>
                    <DATA>
110300003821F64FB80000003800BF193800000052
113010003821F64FB80000003821F64FB800000034
113020003821F64FB80000003880BC81380000004D
...
                    </DATA>
               </FLASHDATA>
```

It is not necessary to include the whole data and code inside the FLASHDATA elements. External source files can be referenced by the filename. This is done for the FLASHDATA at version 'V02'.

```
                    <FLASHDATA ID="F.ECUMEM.MEM.FLASHDATA05.ID" xsi:type="EXTERN-
FLASHDATA">
                         <SHORT-NAME>FLASHDATA_BOOT_V02</SHORT-NAME>
                         <DATAFORMAT SELECTION="INTEL-HEX"/>
                         <DATAFILE LATEBOUND-
DATAFILE="true">FLASHDATA_01.hex</DATAFILE>
                    </FLASHDATA>
                    <FLASHDATA ID="F.ECUMEM.MEM.FLASHDATA06.ID" xsi:type="EXTERN-
FLASHDATA">
```

```
                        <SHORT-NAME>FLASHDATA_CODE_V02</SHORT-NAME>
                        <DATAFORMAT SELECTION="INTEL-HEX"/>
                        <DATAFILE LATEBOUND-
DATAFILE="true">FLASHDATA_02.hex</DATAFILE>
                    </FLASHDATA>
                    <FLASHDATA ID="F.ECUMEM.MEM.FLASHDATA07.ID" xsi:type="EXTERN-
FLASHDATA">
                        <SHORT-NAME>FLASHDATA_DATAA_V02</SHORT-NAME>
                        <DATAFORMAT SELECTION="BINARY"/>
                        <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
                        <DATAFILE LATEBOUND-
DATAFILE="true">FLASHDATA_03.bin</DATAFILE>
                    </FLASHDATA>
                    <FLASHDATA ID="F.ECUMEM.MEM.FLASHDATA08.ID" xsi:type="EXTERN-
FLASHDATA">
                        <SHORT-NAME>FLASHDATA_DATAB_V02</SHORT-NAME>
                        <DATAFORMAT SELECTION="BINARY"/>
                        <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
                        <DATAFILE LATEBOUND-
DATAFILE="true">FLASHDATA_07.bin</DATAFILE>
                    </FLASHDATA>
                </FLASHDATAS>
        </MEM>
```

To advertise the physical memory segments to a programming tool the PHYS-MEM element could be used. All four memory segments of the ECU need to be defined here as shown in Figure 134.

```
        <PHYS-MEM ID="F.ECUMEM.PHYSMEM01.ID">
                <SHORT-NAME/>
                <PHYS-SEGMENTS>
```

Physical memory segment for the flash driver (00000h - 01FFFh):

```
                    <PHYS-SEGMENT ID="F.ECUMEM.PHYSMEM.PHYSSEG01.ID" xsi:type="ADDRDEF-
PHYS-SEGMENT">
                        <SHORT-NAME>BOOT_SEGMENT</SHORT-NAME>
                        <FILLBYTE>FF</FILLBYTE>
                        <BLOCK-SIZE>1024</BLOCK-SIZE>
                        <START-ADDRESS>00</START-ADDRESS>
                        <END-ADDRESS>1FFF</END-ADDRESS>
                    </PHYS-SEGMENT>
```

Physical memory segment for the code (02000h - 1FFFFh):

```
                    <PHYS-SEGMENT ID="F.ECUMEM.PHYSMEM.PHYSSEG02.ID" xsi:type="ADDRDEF-
PHYS-SEGMENT">
                        <SHORT-NAME>CODE_SEGMENT</SHORT-NAME>
                        <FILLBYTE>FF</FILLBYTE>
                        <BLOCK-SIZE>1024</BLOCK-SIZE>
                        <START-ADDRESS>2000</START-ADDRESS>
                        <END-ADDRESS>01FFFF</END-ADDRESS>
                    </PHYS-SEGMENT>
```

Physical memory segment for DataA (20000h - 23FFFh):

```
                    <PHYS-SEGMENT ID="F.ECUMEM.PHYSMEM.PHYSSEG03.ID" xsi:type="SIZEDEF-
PHYS-SEGMENT">
                        <SHORT-NAME>DATAA_SEGMENT</SHORT-NAME>
                        <FILLBYTE>00</FILLBYTE>
                        <BLOCK-SIZE>512</BLOCK-SIZE>
                        <START-ADDRESS>020000</START-ADDRESS>
                        <SIZE>8192</SIZE>
                    </PHYS-SEGMENT>
```

Physical memory segment for the DataB (24000h - 27FFFh):

```
            <PHYS-SEGMENT ID="F.ECUMEM.PHYSMEM.PHYSSEG04.ID" xsi:type="SIZEDEF-
PHYS-SEGMENT">
                  <SHORT-NAME>DATAB_SEGMENT</SHORT-NAME>
                  <FILLBYTE>00</FILLBYTE>
                  <BLOCK-SIZE>512</BLOCK-SIZE>
                  <START-ADDRESS>024000</START-ADDRESS>
                  <SIZE>8192</SIZE>
            </PHYS-SEGMENT>
      </PHYS-SEGMENTS>
</PHYS-MEM>
</ECU-MEM>
<ECU-MEMS>
```

An important branch inside the FLASH container is the ECU-MEM-CONNECTORS. All SESSIONs defined above are referenced here together with a link to the required flash job (DIAG-COMM object).

```
<ECU-MEM-CONNECTORS>
      <ECU-MEM-CONNECTOR ID="F.ECUMEMCON01.ID">
            <SHORT-NAME>FLASH_ECU_Connector</SHORT-NAME>
            <SESSION-DESCS>
```

The first SESSION shall be used for a complete download of all software versions 'V01'. This job also has the highest PRIORITY (0). This means, that this job is executed first, if more than one job is selected.

```
                  <SESSION-DESC DIRECTION="DOWNLOAD">
                        <SHORT-NAME>DOWNLOAD_SOFTWARE_V01_COMPLETE</SHORT-NAME>
                        <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                        <PRIORITY>0</PRIORITY>
                        <SESSION-SNREF SHORT-NAME="FLASH_ECU_HW1_COMPLETE"/>
                        <DIAG-COMM-SNREF SHORT-NAME="DIAGCOMM_FLASH_JOB_01"/>
                        <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true"
IS-AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
                  </SESSION-DESC>
```

The second SESSION is only be used for the DataB memory update to version 'V02'. This requires a different flash-job because data has different security mechanism and only one of the four memory segments need to be programmed. For the update of data memory a lower priority is defined.

```
                  <SESSION-DESC DIRECTION="DOWNLOAD">
                        <SHORT-NAME>DOWNLOAD_UPDATE_DATAB_V02</SHORT-NAME>
                        <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                        <PRIORITY>50</PRIORITY>
                        <SESSION-SNREF SHORT-NAME="UPDATE_DATAB"/>
                        <DIAG-COMM-SNREF SHORT-NAME="DIAGCOMM_FLASH_JOB_02"/>
                        <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true"
IS-AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
                  </SESSION-DESC>
```

The update of flash driver and code memory with version 'V02' requires a further flash-job.

```
                  <SESSION-DESC DIRECTION="DOWNLOAD">
                        <SHORT-NAME>DOWNLOAD_UPDATE_BOOT_AND_CODE_V02</SHORT-
NAME>
                        <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                        <PRIORITY>10</PRIORITY>
                        <SESSION-SNREF SHORT-NAME="UPDATE_BOOT_AND_CODE_HW1"/>
                        <DIAG-COMM-SNREF SHORT-NAME="DIAGCOMM_FLASH_JOB_03"/>
                        <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true"
IS-AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
                  </SESSION-DESC>
```

The download of DataA and DataB segment requires a new job too.

```
                    <SESSION-DESC DIRECTION="DOWNLOAD">
                          <SHORT-NAME>DOWNLOAD_UPDATE_DATAA_AND_DATAB_V02</SHORT-
NAME>
                          <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                          <PRIORITY>40</PRIORITY>
                          <SESSION-SNREF SHORT-NAME="UPDATE_DATA_HW2_COMPLETE"/>
                          <DIAG-COMM-SNREF SHORT-
NAME="DIAGCOMM_FLASH_ECU_JOB_04"/>
                          <AUDIENCE IS-MANUFACTURING="true" IS-SUPPLIER="true"
IS-AFTERSALES="true" IS-DEVELOPMENT="true" IS-AFTERMARKET="true"/>
                    </SESSION-DESC>
             </SESSION-DESCS>
             <ECU-MEM-REF ID-REF="F.ECUMEM01.ID" DOCTYPE="FLASH"/>
             <LAYER-REFS>
                    <LAYER-REF DOCTYPE="LAYER" DOCREF="ECUVARIANT01" ID-
REF="ECUVARIANT01.ID"/>
             </LAYER-REFS>
       </ECU-MEM-CONNECTOR>
</ECU-MEM-CONNECTORS>
</FLASH>
```

Flash jobs can be reused in SESSION-DESCs if the sequence of diagnostic services and the security mechanisms for up- or download in the respective SESSIONs is equal.
It is also possible to describe multiple processor ECUs within one flash container. If the different memory areas of the processors inside the ECU can be programmed by using one continuos address space all SESSIONs, DATABLOCKs, FLASHDATAs and PHYSMEMs can be defined in only one ECU-MEM object.
To support more than one ECU with different physical memory structures or ECUs with multiple processors and overlapping memory areas multiple ECU-MEMs and ECU-MEM-CONNECTORS shall be defined in one flash container instance.

7.6.1 OVERVIEW

In the automotive electronics industry, the term “Variant Coding” describes a procedure that allows the ECU software to be adapted to a vehicle-specific (optional components, equipments and instrumentation) or a localisation specific (e.g. country) environment. “Variant Coding” of an ECU is typically performed during vehicle or ECU production, in the aftermarket (upgrading vehicle equipment) and in a service environment (replacement of defective parts or equipment). For the purpose of this document, the term “ECU software configuration” or “ECU configuration” in brief will be used to describe this process, because this term describes more accurately what is done during this procedure. To this end, the ECU configuration part of ODX provides a set of data elements that allow modifying subsections of the ECU's memory using diagnostic communication i.e. diagnostic services. That means ODX contains on one hand *predefined data sets* that can be directly used for configuration as well as the description of their *substructure and conversion methods* in order to change parts of the data dynamically during runtime of the diagnostic system driven by the ODX data.
The following use cases are typically handled via ECU configuration and are covered by the ODX data model:

- Adjustment of an ECU according to a specific vehicle environment
- Country specific configuration
- Setting of characteristic curves
- Enabling/Disabling of optional functionality

Even though the most obvious use case for the described model is to *write* configuration data *to* an ECU, also *reading* of configuration data *from* an ECU is supported.

7.6.2 GENERAL MODELING CONCEPTS

The central element of the ODX ECU configuration concept is the CONFIG-RECORD. A CONFIG-RECORD object acts as a notional memory buffer, which corresponds to a contiguous section of the ECU's permanent memory. A CONFIG-RECORD connects the data and their description on the one hand and the ECU's permanent memory and the diagnostic services to manipulate these data on the other hand. Like most other ODX data, configuration data can also be described in either physical or coded representation or even a combination of both.
Each CONFIG-RECORD describes its own notional buffer with a fictive start address of zero. The address in the ECU's permanent memory is defined at DIAG-COMM-DATA-CONNECTOR/WRITE-SERVICE/PHYS-CONSTANT-VALUE[7] which can also be used for definition of DIDs (data identifier like local ID) if appropriate.
Several CONFIG-RECORDs can be aggregated into a CONFIG-DATA element. The CONFIG-DATA is considered to contain the Configuration data description for a whole ECU. Several CONFIG-DATAs can be aggregated in an ECU-CONFIG instance. ECU-Config is a separate ODX CATEGORY (like FLASH or DIAG-LAYER-CONTAINER) [8].

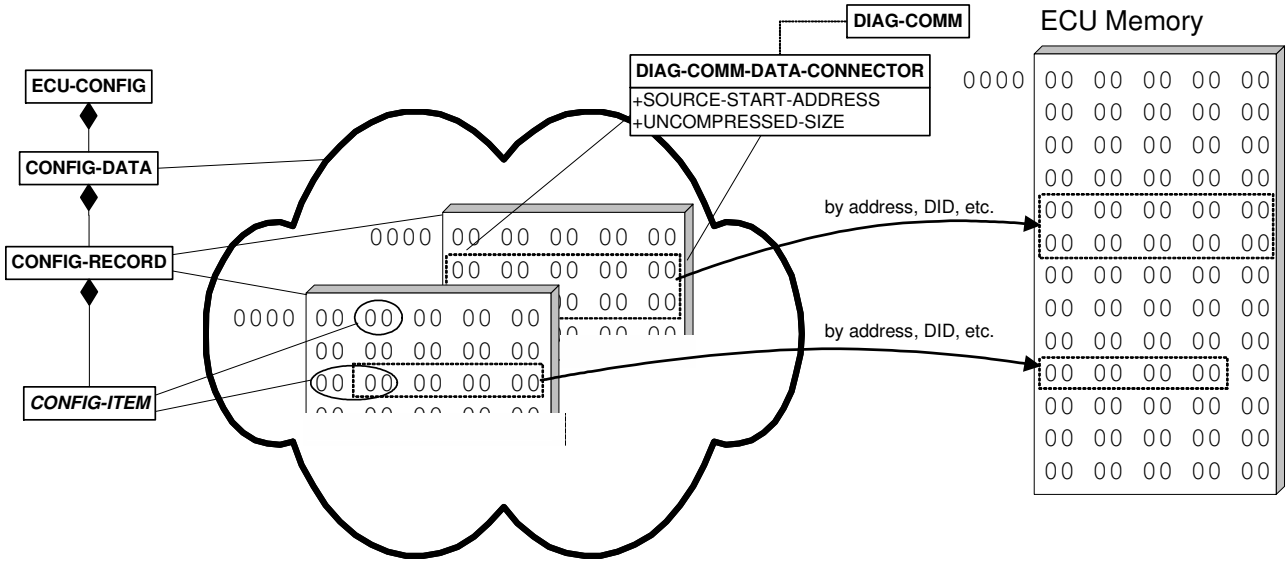


Figure 135 — ECU configuration overview[9)]

7) The same applies for READ-SERVICE.

8) Please note that the aggregation of CONFIG-RECORD objects within a CONFIG-DATA object does *not* convey any further meaning such as ordering of the notional memory buffers or read/write sequence.

9) The UML model elements in this figure are simplified. Please see section 7.7.3 for the comprehensive model.

7.7 ODX DATA MODEL FOR ECU CONFIGURATION

7.7.1 DESCRIPTION OF THE ECU CONFIGURATION DATA MODEL

7.7.1.1 OVERVIEW

The data for ECU configuration can be described in two different ways:

- Blocks of binary data (DATA-RECORD)
- Fine grained data items that correspond to single functions of the ECU (CONFIG-ITEM, described by ITEM-VALUE if value can be changed at runtime)

7.7.1.2 DESCRIBING BLOCKS OF BINARY DATA

A CONFIG-RECORD object can optionally contain one or many DATA-RECORD objects. A DATA-RECORD object is a contiguous block of binary data. Each DATA-RECORD object can be identified by its IDENT-VALUE (element name: DATA-ID). Each DATA-RECORD object represents *one alternative content* of the CONFIG-RECORD object it belongs to. The decision about which DATA-RECORD to use in an actual ECU configuration event has to be handled outside of ODX.

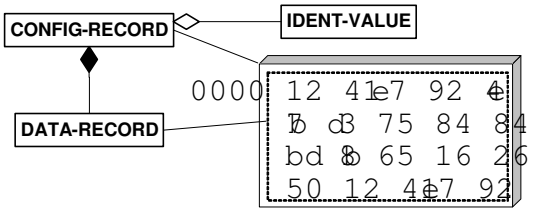


Figure 136 — Using blocks of binary data via DATA-RECORD[10)]

A DATA-RECORD can only describe coded values. The conversion of these binary data to physical values (and vice versa) is possible by using the fine grain ECU configuration data description by the CONFIG-ITEM objects (see next paragraph).
In a runtime environment, a user may be given the choice to select one of the DATA-RECORDs defined in the respective ODX file by selecting its IDENT-VALUE (DATA-ID) and to use this block data in an ECU configuration programming event.

7.7.1.3 DESCRIBING INDIVIDUAL CONFIGURATION ITEMS

A CONFIG-RECORD object may optionally be completely described via so-called CONFIG-ITEM elements. A CONFIG-ITEM may be thought of as a bit or a number of contiguous bits, enabling or disabling a specific ECU function or setting a value for use by some ECU function(s). The entirety of the notional memory may be defined by CONFIG-ITEM[11)] elements. CONFIG-ITEMS also relay to DATA-OBJECT-PROPERTIES (DOPs) i.e. in a runtime environment an external process may supply a physical values for each of the CONFIG-ITEM elements which will be converted into the coded representation

10) The UML model elements in this figure are simplified. Please see section 7.7.3 for the comprehensive model. For more information on how to specify binary data blocks using the DATA element see also section 6.5.2.4 (FLASHDATA).

11) CONFIG-ITEM is a generalization of some specialised model elements. See Figure 141 — ECU configuration items for details.

defining a spot in the notional memory buffer. Alternatively a CONFIG-ITEM element can also have a PHYSICAL-DEFAULT-VALUE, which will be used in case no actual physical value is being supplied. Anyway the section of a DATA-RECORD covered by a CONFIG-ITEM will be changed. These sections must not overlap.

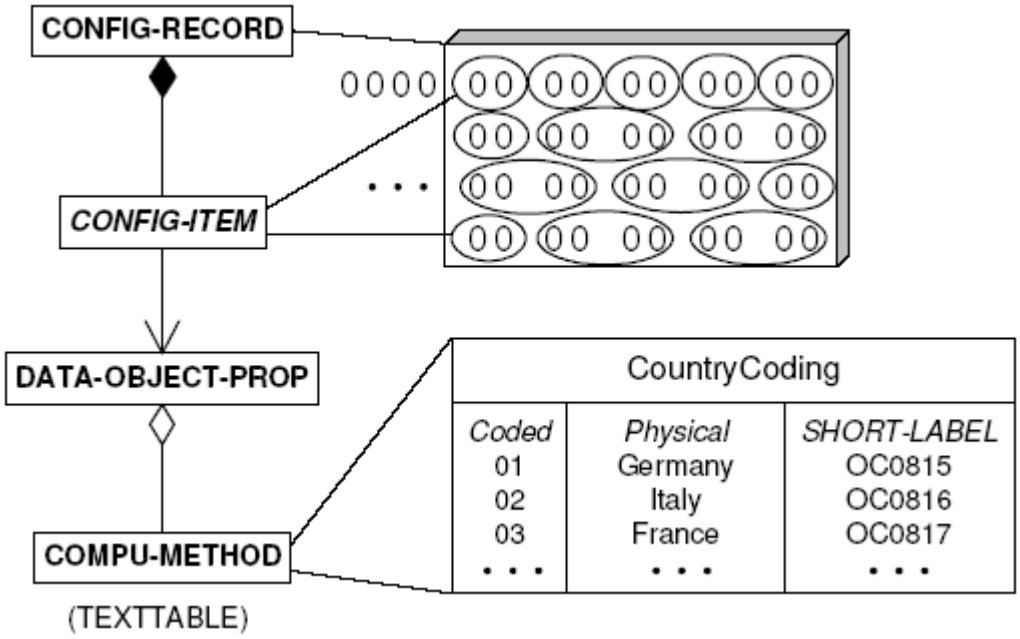


Figure 137 — Using CONFIG-ITEMs[12)]

The are four specialisations of CONFIG-ITEM:

— OPTION-ITEM
This describes a e.g. function related configuration element of the ECU. The majority of CONFIG-ITEMs will be of this type. If for a certain use-case option codes are required to identify the various configuration options, this option code will be put in the SHORT-LABEL element of the corresponding COMPU-SCALE of the TEXT-TABLE being used as COMPU-METHOD in the DOP that is linked to the CONFIG-ITEM.

— SYSTEM-ITEM
This describes an instruction to the runtime system to write a data element (e.g. the tester ID or the current data) to the ECU's programmable memory. This is in analogy to the SYSTEM parameter, which is described in section 6.3.5.4. See also table 3 for a list of predefined SYSPARAM values.

— CONFIG-ID-ITEM
There exists at most one CONFIG-ID-ITEM for every CONFIG-RECORD, which is used to define how the *identification* of that CONFIG-RECORD (given by its IDENT-VALUE/CONFIG-ID) is being located in the notional memory buffer. Using this method the identification can be written to the ECUs memory. When reading data from the ECU, this can be used to identify the matching CONFIG-RECORD in the ODX data.

— DATA-ID-ITEM
There exists at most one DATA-ID-ITEM per DATA-RECORD, which is used to define how the identification of the selected DATA-RECORD (given by its IDENT-VALUE/DATA-ID) is being converted to or from the notional memory buffer. Using this method the

12) The UML model elements in this figure are simplified. Please see section 7.7.3 for the comprehensive model.

identification can be written to the ECUs memory. When reading data from the ECU, this can be used to identify the matching DATA-RECORD in the ODX data.

7.7.1.4 DESCRIPTION OF CONFIGURATION VALUES

An Alternative to the usage of a TEXTTABLE for the description of configuration values is the definition of ITEM-VALUES. They allow to specify the potential values of an OPTION-ITEM with some additional information:

- PHYSICAL-CONSTANT-VALUE: contains the value to be converted by the related DOP
- MEANING: a short description of the consequences to the ECU's functionality if this value is selected
- KEY: a unique key value to identify this ITEM-VALUE
- RULE: a placeholder for a regular expression to determine if this value is valid for a certain vehicle configuration
- DESCRIPTION: Offers an additional opportunity to describe the functionality of this certain value if selected

Note that the PHYSICAL-CONSTANT-VALUE is also mapped against the DOP referenced by the OPTION-ITEM.

7.7.1.5 COMBINING BINARY DATA BLOCKS WITH ECU FUNCTIONS

The two methods described above can be combined in a straightforward manner as shown below. The binary data supplied by one of the DATA-RECORD object will initialise the notional memory buffer in a first step. Then one or more of the CONFIG-ITEM (type OPTION-ITEM) elements of the CONFIG-RECORD together with some PHYSICAL-DEFAULT-VALUEs or user supplied values can modify the memory buffer under control of a user.

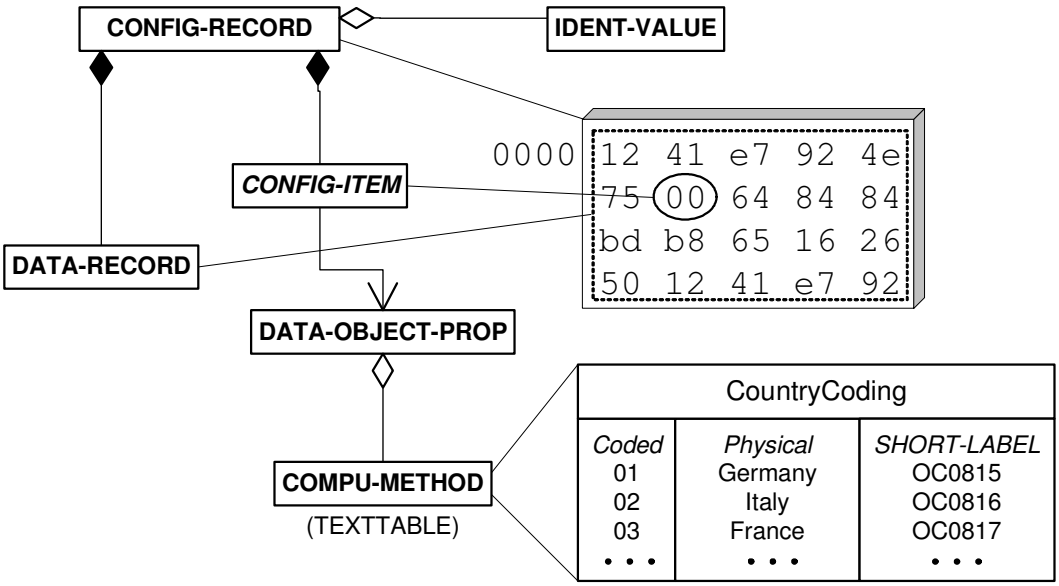


Figure 138 — Combining CONFIG-ITEM and DATA-RECORD[13)]

13) The UML model elements in this figure are simplified. Please see section 7.7.3 for the comprehensive model.

7.7.2 READING AND WRITING CONFIGURATION DATA FROM AND TO THE ECU

Figure 139 shows the general idea how to write configuration data to an ECU using a KWP2000 WriteMemoryByAddress service as an example. First, a runtime environment needs to identify the services to use for reading and writing configuration data. This is accomplished by the DIAG-COMM-DATA-CONNECTOR element of the CONFIG-RECORD. The DIAG-COMM-DATA-CONNECTOR references via ODX-link an optional WRITE-SERVICE to be used to write configuration data to the ECU. The IN-PARAM-IF link at WRITE-SERVICE-VALUE of the DIAG-COMM-DATA-CONNECTOR identifies the parameter of the write service, which holds the configuration data. The attribute PHYS-CONSTANT-VALUE of the WRITE-SERVICE-VALUE is supposed to contain the ECU addressing information in a general sense. It is not limited to just memory addresses it could also be a DID number, PID number or similar. A hexadecimal representation is assumed though. The attribute SEMANTIC can be set to ADDRESS or DID.

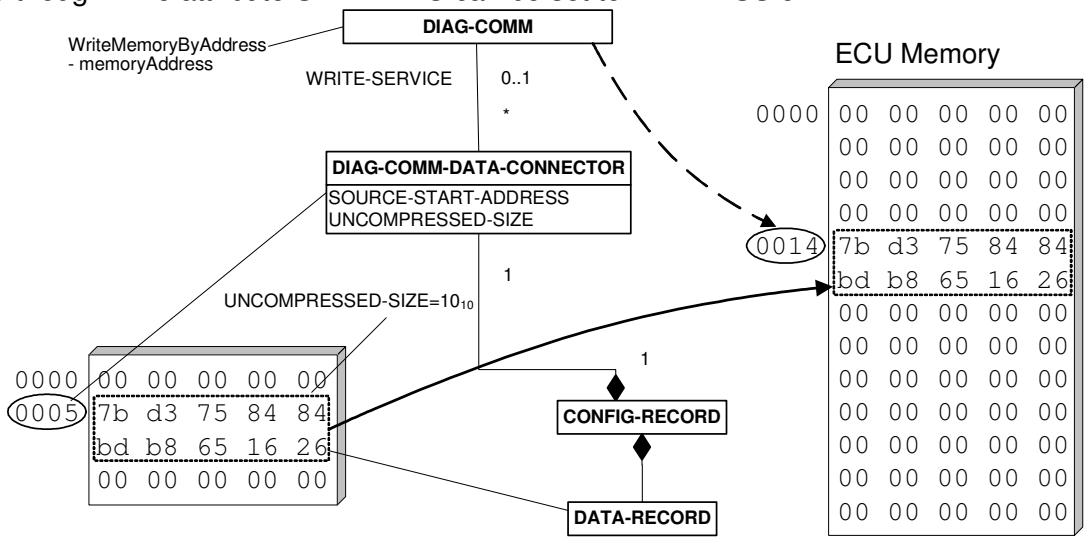


Figure 139 — Write data to the ECU by KWP2000-WriteMemoryByAddress[14)]

Accordingly, DIAG-COMM-DATA-CONNECTOR also refers to a READ-SERVICE and an OUT-PARAM-IF for reading configuration data from an ECU.
In the event, that the WRITE-SERVICE cannot transfer the complete CONFIG-RECORD in one single programming event, the CONFIG-RECORD can be subdivided into several DIAG-COMM-DATA-CONNECTORS. For the purpose of ECU configuration, several DIAG-COMM-DATA-CONNECTORSs are used solely to split up a CONFIG-RECORD into smaller chunks of data, that can be used as a PARAM in the PDU of the read or write service. For that reason the order of the DIAG-COMM-DATA-CONNECTORSs has to be defined.
The SOURCE-START-ADDRESS identifies the address of the first byte of the datablock in the notional memory buffer. The length of the datablock is given by the UNCOMPRESSED-SIZE element.

14) The UML model elements are simplified in this figure. Please see section 7.7.3 for the comprehensive model.

7.7.3 COMPREHENSIVE UML MODEL

The following class diagrams show the structures that ODX provides for the purpose of ECU configuration. The ODX data element ECU-CONFIG of type ODX-CATEGORY aggregates all data elements for ECU configuration programming. Being of type ODX-CATEGORY means, that ECU-CONFIG is a separate XML-file of the ODX XML files family like COMPARAM-SPEC, DIAGLAYER-SPEC, FLASH etc. with the postfix .odx-e or simply .odx. Inheritance from CATEGORY means also that authoring and versioning information can be applied (COMPANY-DATA, ADMIN-DATA).

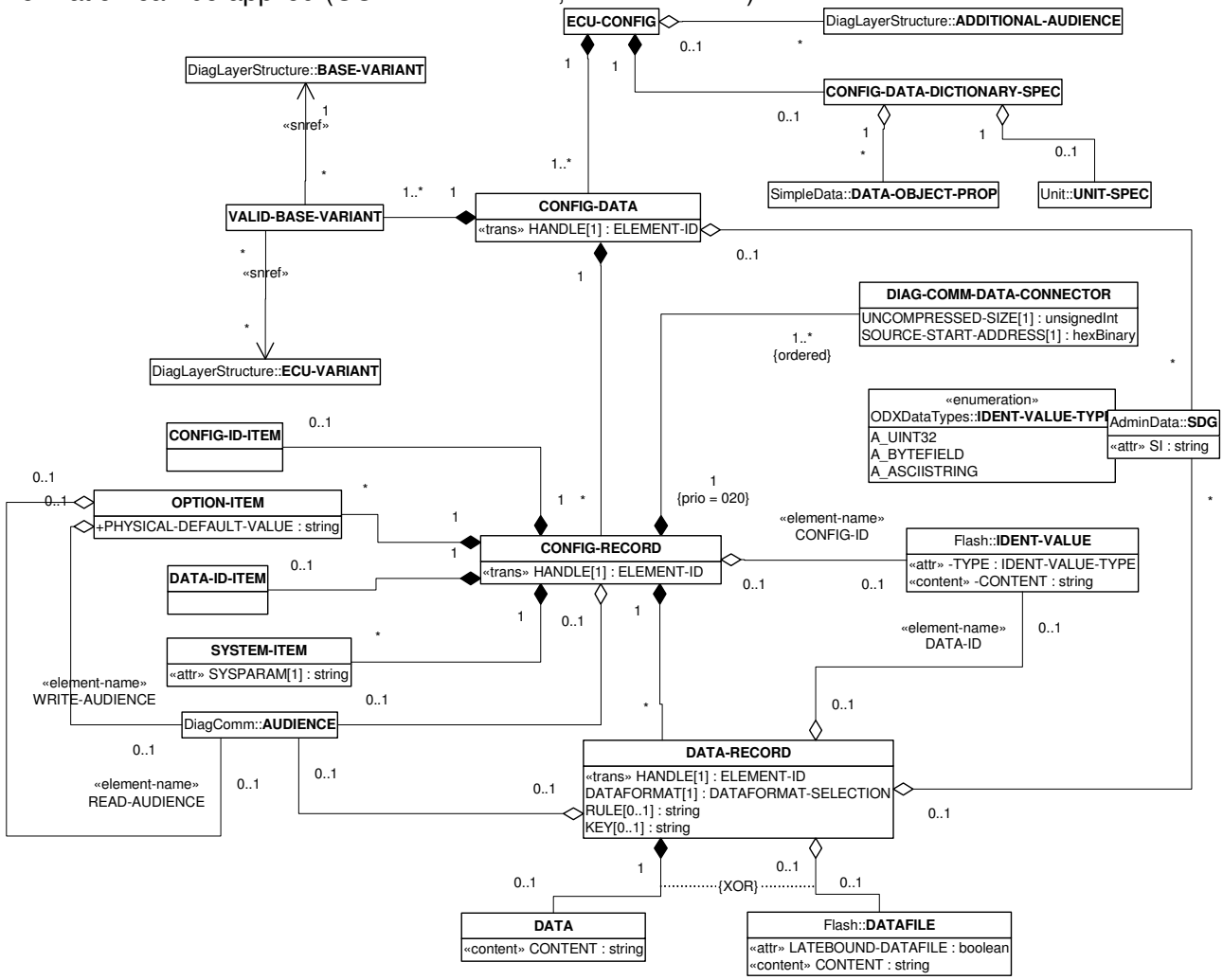


Figure 140 — ECU configuration data

7.7.4 DRAWING ECUCONFIG

The meaning of CONFIG-DATA is described in the beginning of this chapter. The VALID-BASE-VARIANT determines the Base-Variant and ECU-Variants the complete set of CONFIG-DATA is valid for. Only one BASE-VARIANT must be given. In the most simple case nothing more is defined and the CONFIG-DATA is valid for this BASE-VARIANT and all ECU-VARIANTs that inherit from that BASE-VARIANT. Not that the validity can be even more restricted or filtered by the RULE and KEY attributes of DATA-RECORD and ITEM-VALUE.

If some ECU-VARIANTS (of that BASE-VARIANT) are explicitely defined the validity of CONFIG-DATA is restricted to them.
Each CONFIG-DATA contain several CONFIG-RECORDs. Each CONFIG-RECORD can contain several DATA-RECORD which are predefined blocks of coding information. KEY and RULE can be used for filtering of data. For example in RULE a set of numbers can be defined like “K1 K4 K6” (means logical or) and if K1 (Konfiguration 1) is given only CONFIG-RECORD suitablefor that K1 could be used. Note that the format for this is not standardized yet. The same applies to KEY.
The attribute DATAFORMAT of type DATAFORMAT-SELECTION can be either INTEL-HEX, MOTOROLA-S or BINARY (compare FLASHDATA/DATAFORMAT). DATAFILE is described at the FLASH section whereas DATA corresponds to INTERN-FLASHDATA.
Each DATA-RECORD can have a IDENT-VALUE (element name DATA-ID) where the IDENT-VALUE-TYPE defines the datatype (see description of FLASH) and CONTENT the matching value. The position of this ID information in the CONFIG-RECORD is defined by the DATAID-ITEM component of CONFIG-RECORD. The same applies to IDENT-VALUE/CONFIG-ID and CONFIG-ID-ITEM. Each CONFIG-ITEM refers to an DATA-OBJECT-PROPERTY that defines transformation from physical to coded representation and vice versa.

Drawing: ConfigItem

Package: EcuConfig

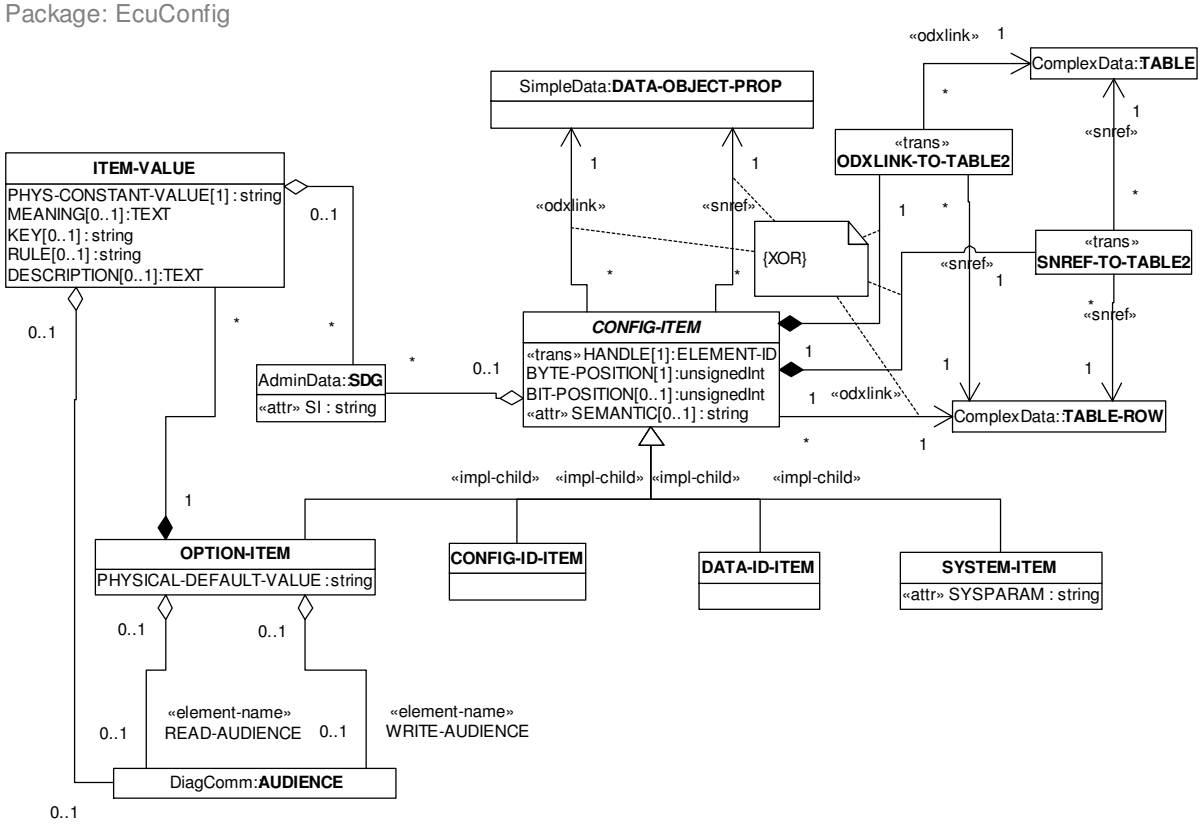


Figure 141 — ECU configuration items

As an alternative to the definition of ITEM-VALUEs and an appropriate DOP reference a TABLE or a TABLE-ROW may be referenced. In this case the possible values for a CONFIG-ITEM are defined in the PARAMETERs contained in the TABEL-ROW's underlying STRUCTURE.

Since every PARAMETER in a STRUCTURE may refer to a different DOP, it is possible to have different conversions for each potential value of the CONFIG-ITEM. In order to be able to calculate the right UNCOMPRESSED-SIZE of a DIAG-COMM-DATA-CONNECTOR related to the CONFIG-RECORD containing this CONFIG-ITEM, it is required that the DOPs use the same DIAG-CODED-TYPE.

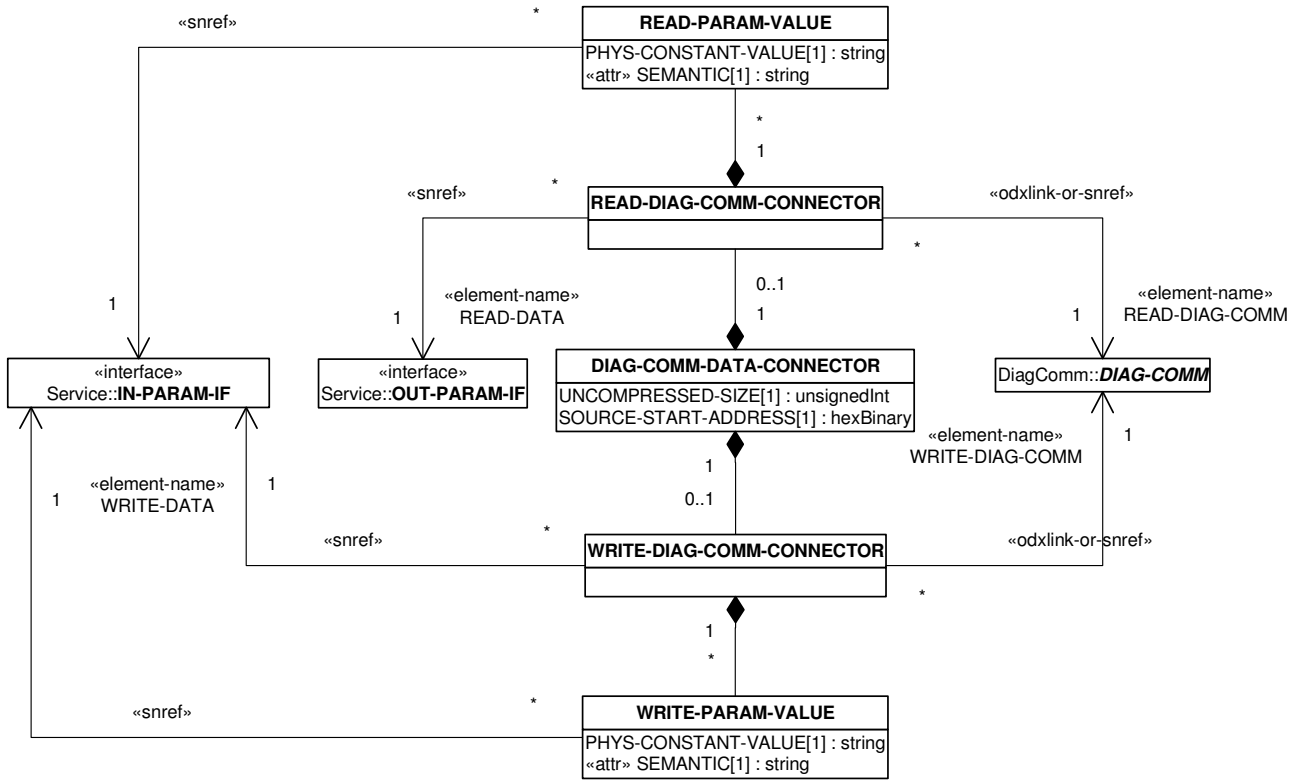


Figure 142 — ECU configuration: DIAG-COMM-DATA-CONNECTOR

The DIAG-COMM-DATA-CONNECTOR (fig.7) defines how to read and write the data described in a CONFIG-RECORD. In the WRITE case a DIAG-SERVICE or SINGLE-ECU-JOB is defined that must be used for writing the data of the CONFIG-RECORD. The reference from WRITE-PARAM-VALUE to an IN Parameter of the DIAG-COMM determines the PARAM that is used e.g. for the memory address or the DID information (e.g. local ID LID). The reference from WRITE-DIAG-COMM-CONNECTOR to an IN Parameter of the DIAG-COMM with element name WRITE-DATA determines the data block parameter where the data of CONFIG-DATA will start.
Almost the same mechanism applies for the read use case, except that the READ-DATA element points to a response parameter (via OUT-PARAM-IF).

7.7.5 EXAMPLE

The following piece of XML shows how a DATA-RECORD and a CONFIG-ITEM look like in ECU-CONFIG.

```
<ECU-CONFIG ID="000001">
      <SHORT-NAME>ECU_1</SHORT-NAME>
      <CONFIG-DATAS>
            <CONFIG-DATA>
```

```
                    <VALID-BASE-VARIANTS>
                           <VALID-BASE-VARIANT>
                                  <LAYER-REF DOCREF="ECU1" DOCTYPE="CONTAINER" ID-
REF="ECU1"/>
                           </VALID-BASE-VARIANT>
                    </VALID-BASE-VARIANTS>
                    <SHORT-NAME>CONFIG_DATA_1</SHORT-NAME>
                    <CONFIG-RECORDS>
                           <CONFIG-RECORD>
                                  <SHORT-NAME>Kodierung_gesamt</SHORT-NAME>
                                  <DIAG-COMM-DATA-CONNECTORS>
                                         <DIAG-COMM-DATA-CONNECTOR>
                                                <SOURCE-START-ADDRESS>0</SOURCE-
START-ADDRESS>
                                                <UNCOMPRESSED-SIZE>1</UNCOMPRESSED-
SIZE>
                                         </DIAG-COMM-DATA-CONNECTOR>
                                  </DIAG-COMM-DATA-CONNECTORS>
                                  <CONFIG-ID TYPE="A_UINT32">01</CONFIG-ID>
                                  <DATA-RECORDS>
                                         <DATA-RECORD>
                                                <SHORT-NAME>Default_String</SHORT-
NAME>
                                                <DATA-ID
TYPE="A_ASCIISTRING">0001</DATA-ID>
                                                <DATAFORMAT SELECTION="BINARY"/>

      <DATA>01010101010101010101010101010101</DATA>
                                         </DATA-RECORD>
                                  </DATA-RECORDS>
                                  <OPTION-ITEMS>
                                         <OPTION-ITEM>
                                                <SHORT-NAME>Config_ITEM_1</SHORT-
NAME>
                                                <BYTE-POSITION>0</BYTE-POSITION>
                                                <BIT-POSITION>0</BIT-POSITION>
                                                <ITEM-VALUES>
                                                       <ITEM-VALUE>
                                                              <PHYSICAL-DEFAULT-
VALUE>0</PHYSICAL-DEFAULT-VALUE>

      <MEANING>Off</MEANING>
                                                                     <KEY>0001</KEY>

                                                       </ITEM-VALUE>
                                                       <ITEM-VALUE>
                                                              <PHYSICAL-DEFAULT-
VALUE>1</PHYSICAL-DEFAULT-VALUE>
                                                              <MEANING>on</MEANING>
                                                              <KEY>0002</KEY>
                                                       </ITEM-VALUE>
                                                </ITEM-VALUES>
                                         </OPTION-ITEM>
                                  </OPTION-ITEMS>
                           </CONFIG-RECORD>
                    </CONFIG-RECORDS>
             </CONFIG-DATA>
      </CONFIG-DATAS>
</ECU-CONFIG>
```

7.8 FUNCTION DICTIONARY

7.8.1 INTENTION

ODX provides in a wide range a communication-oriented view on an ECU's diagnostic functionality. This does not meet today's car designs in certain aspects, because many functions of a car are distributed across several ECUs. The function-oriented point of view to diagnostics is covered with the structures described in this chapter in addition to the aspects of communication-oriented diagnostics Definitions and Requirements.

7.8.2 DEFINITIONS AND REQUIREMENTS

7.8.2.1 FUNCTION

A function in ODX represents diagnostic data from a specific car's function-oriented point of view.

For example the function "Blinking left" is implemented across different ECUs and related to fault memories, input/output controls, measurement values and so on. Furthermore the function "Blinking left" may be valid for several model lines and model years. From a functional point of view it is interesting to know which ECUs and diagnostic elements of an ECU are part of a function. Finally, a function may have input and output parameters to influence the behaviour of a function respectively indicating the correct function behaviour.

Functions are defined in hierarchical structures (by recursion) to allow to express different functional granularities.

- Blinking
  - Blinking left
  - Blinking right
  - Warning lights
  - …

7.8.2.2 FUNCTION GROUP

A Function Group is a container for already defined functions regardless to their hierarchical context. The Function Group "Lights" could for example contain the Functions "Blinking" and "Illumination".

- Lights
  - Illumination
    - Head lights
    - …
  - Blinking
    - Blinking left
    - Blinking right
    - Warning lights
    - …

A Function Group may also contain other Function Groups. Next to that, the same requirements as for a FUNCTION (relation to ECUs, services, DTCs, parameters…) apply.

7.8.2.3 SUBCOMPONENT

A subcomponent is considered to be a physical unit of an ECU that covers certain additional functionality (e.g. a LIN-Slave). In opposite to a FUNCTION or a FUNCTION-GROUP a subcomponent is always related to one explicit ECU (or even ECU-VARIANT). The SUBCOMPONENT's description is needed within the description of communicational behaviour (i.e. inside a DIAG-LAYER).
Therefore the Function Dictionary model in ODX is divided into two parts:

- A new ODX CATEGORY to cover the FUNCTION/FUNCTION-GROUP use cases
- An extension to DIAG-LAYER to cover the SUBCOMPONENT use case

7.8.2.4 ODX CATEGORY FUNCTION-DICTIONARY

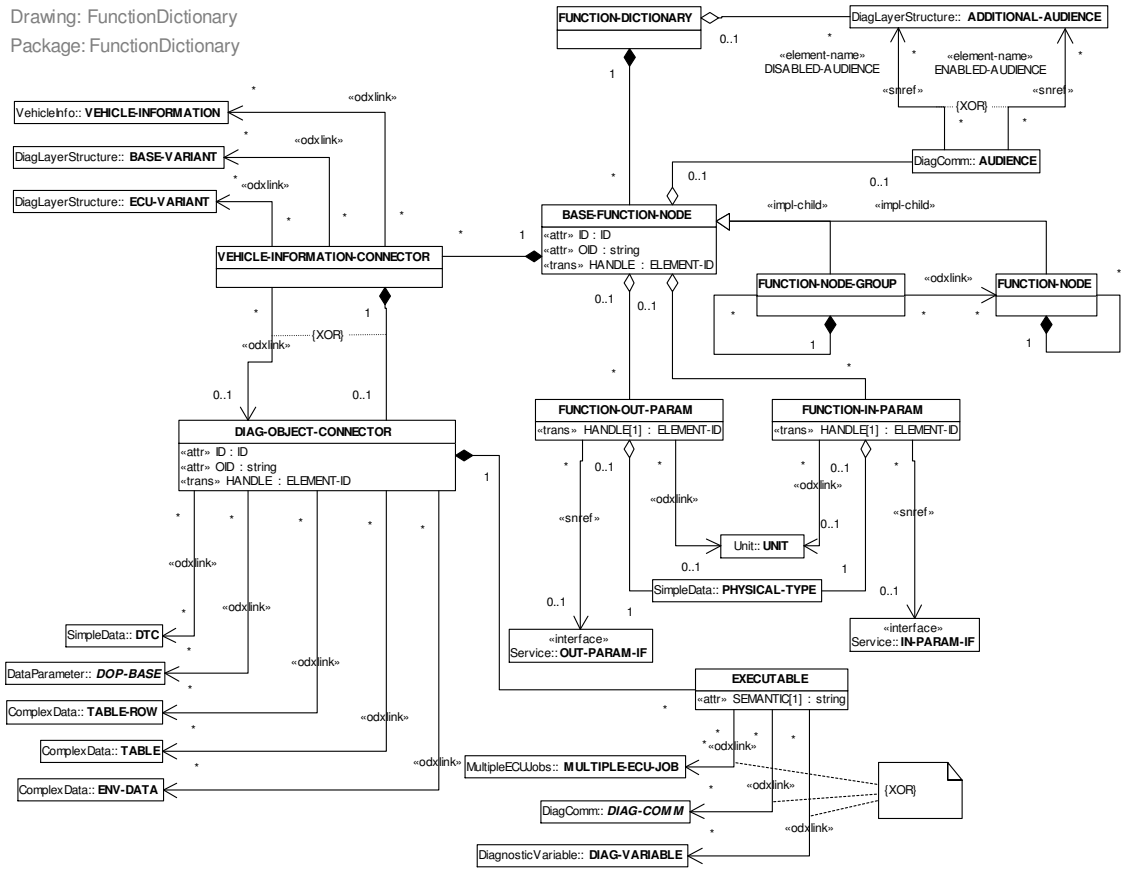


Figure 143 — FUNCTION-DICTIONARY

FUNCTION DICTIONARY inherits from CATEGORY and provides ADMIN-DATA and COMPANY-DATA by default.

7.8.2.5 BASE- FUNCTION-NODE

A BASE-FUNCTION-NODE aggregates all common features of a Function/Function-Group (see previous chapter) and therefore is implemented in two ways:

- FUNCTION-NODE
- FUNCTION-NODE-GROUP

The Element FUNCTION-NODE represents a FUNCTION as part of a function hierarchy.

The requirement of grouping functions is implemented by a FUNCTION-NODE-GROUP. FUNCTION-NODES are referenced via ODX link by a FUNCTION-NODE-GROUP.

7.8.2.5.1 VEHICLE-INFORMATION-CONNECTOR

Via an VEHICLE-INFORMATION-CONNECTOR a BASE-FUNCTION-NODE references a BASE-VARIANT or ECU-VARIANT from the DIAG-LAYER-CONTAINER. Additionally a VEHICLE-INFORMATION may be referenced to express for which car the FUNCTION is designed.

7.8.2.5.2 DIAG-OBJECT-CONNECTOR

References to diagnostic objects in a DIAG-LAYER-CONNECTOR (e.g. DTCs and TABLEs, DIAG-COMMs etc).are provided via the DIAG-OBJECT-CONNECTOR.

7.8.2.6 EXECUTABLE

A Function may be “executable” by referring to a SINGLE-ECU-JOB, a DIAG-SERVICE or a DIAG-VARIABLE via an ODX-Link.

7.8.2.7 FUNCTION-IN-PARAM/FUNCTION-OUT-PARAM

A Function or Function Group may have input and output parameters. These are represented by the FUNCTION-IN-PARAM and the FUNCTION-OUT-PARAM. Both have a PHYSICAL-TYPE and a UNIT. Optionally they may refer to a Parameter of an EXECUTABLE (a SINGLE-ECU-JOB, a DIAG-SERVICE or a DIAG-VARIABLE) via IN-PARAM-IF/ OUT-PARAM-IF.

7.8.3 USAGE SCENARIO (FUNCTION-DICTIONARY-SPEC)

This is an usage scenario for a function called “Blinking left” contained in a function group called “All Blinking Functions”.

```
<FUNCTION-DICTIONARY-SPEC ID="X0001">
      <SHORT-NAME>FUNCDIC_Platform1</SHORT-NAME>
```

At first the function “Blinking left” with SHORT-NAME and DESCRIPTION is implemented:

```
<FUNCTION-NODES>
      <FUNCTION-NODE ID="FNC_Blinking-left">
            <SHORT-NAME>FncBlinkingLeft</SHORT-NAME>
            <LONG-NAME>Function blinking left</LONG-NAME>
            <DESC><p>Periodic on/off lights left</p></DESC>
```

A Function may be related to one or more ECUs/ Variants via the VEHICLE-INFORMATION-CONNECTORs. The relation to a DIAL-LAYER-CONTAINER is established via a BASE-OR-ECU-VARIANT-REF to the corresponding element in DIAG-LAYER-CONTAINER (alternatively the BASE-OR-ECU-VARIANT-SNREF can be used).

```
<VEHICLE-INFORMATION-CONNECTORS>
      <VEHICLE-INFORMATION-CONNECTOR ID="Vhc_FrontEcu">
            <SHORT-NAME>Vhc_FrontEcu</SHORT-NAME>
            <LONG-NAME>All diagnostic elements on Front ECU for function
Blinking left</LONG-NAME>
```

```
            <BASE-VARIANTS>
                  <BASE-VARIANT ID-REF="EV_Front"/>
            <BASE-VARIANTS>
```

The information which elements of the corresponding DIAG-LAYER-CONTAINER element are related to this function is stored in the element DIAG-LAYER-CONNECTOR. In this example DTCs and DIAG-COMMs are referenced.

```
            <DIAG-OBJECT-CONNECTOR ID="dlc_1">
                  <SHORT-NAME>sdfsdf</SHORT-NAME>
                  <DTCS>
                        <DTC-REF ID-REF="DTC_Pin2-NoSignal"/>
                        <DTC-REF ID-REF="DTC_Pin23-NoSignal"/>
                        …
                  </DTCS>
                  <DIAG-COMMS>
                  <DIAG-COMM-REF ID-REF="DiagService_IOCPin_1"/>
                  <DIAG-COMM-REF ID-REF="DiagService_IOCPin_2"/>
                  </DIAG-COMMS>
            </DIAG-OBJECT-CONNECTOR>
```

Finally there is the optional opportunity to execute the function. Therefore an EXECUTABLE with a reference to a service may be used.

```
            <EXECUTABLE>
                  <DIAG-COMM-REF ID-REF="SingleEcuJob_TestAllPins"/>
            <EXECUTABLE>
      </VEHICLE-INFORMATION-CONNECTOR>
```

An instance of a VEHICLE-INFORMATION-CONNECTOR for a second ECU is shown below.
The element FUNCTION-NODE-GROUP is used to aggregate already implemented functions. This may be used as a convenience function or a shortcut.

```
      <FUNCTION-NODE-GROUPS>
            <FUNCTION-NODE-GROUP ID="FG_AllBlinking">
                  <SHORT-NAME>FG_AllBlinking</SHORT-NAME>
                  <LONG-NAME>All blinking functions</LONG-NAME>
                  <FUNCTION-NODE-REFS>
                        <FUNCTION-NODE-REF ID-REF="FncBlinkingLeft"/>
                  </FUNCTION-NODE-REFS>
            </FUNCTION-NODE-GROUP>
      </FUNCTION-NODE-GROUPS>


            <DIAG-OBJECT-CONNECTOR ID="dlc_1">
                  <SHORT-NAME>sdfsdf</SHORT-NAME>
                  <DTCS>
                        <DTC-REF ID-REF="DTC_Pin2-NoSignal"/>
                        <DTC-REF ID-REF="DTC_Pin23-NoSignal"/>
                        …
                  </DTCS>
                  <DIAG-COMMS>
                  <DIAG-COMM-REF ID-REF="DiagService_IOCPin_1"/>
                  <DIAG-COMM-REF ID-REF="DiagService_IOCPin_2"/>
                  </DIAG-COMMS>
            </DIAG-OBJECT-CONNECTOR>
```

Finally there is the optional opportunity to execute the function. Therefore an EXECUTABLE with a reference to a service may be used.

```
            <EXECUTABLE>
                  <DIAG-COMM-REF ID-REF="SingleEcuJob_TestAllPins"/>
            <EXECUTABLE>
      </VEHICLE-INFORMATION-CONNECTOR>
```

An instance of a VEHICLE-INFORMATION-CONNECTOR for a second ECU is shown below.
The element FUNCTION-NODE-GROUP is used to aggregate already implemented functions. This may be used as a convenience function or a shortcut.

```
<FUNCTION-NODE-GROUPS>
      <FUNCTION-NODE-GROUP ID="FG_AllBlinking">
            <SHORT-NAME>FG_AllBlinking</SHORT-NAME>
            <LONG-NAME>All blinking functions</LONG-NAME>
            <FUNCTION-NODE-REFS>
                  <FUNCTION-NODE-REF ID-REF="FncBlinkingLeft"/>
            </FUNCTION-NODE-REFS>
      </FUNCTION-NODE-GROUP>
</FUNCTION-NODE-GROUPS>
```

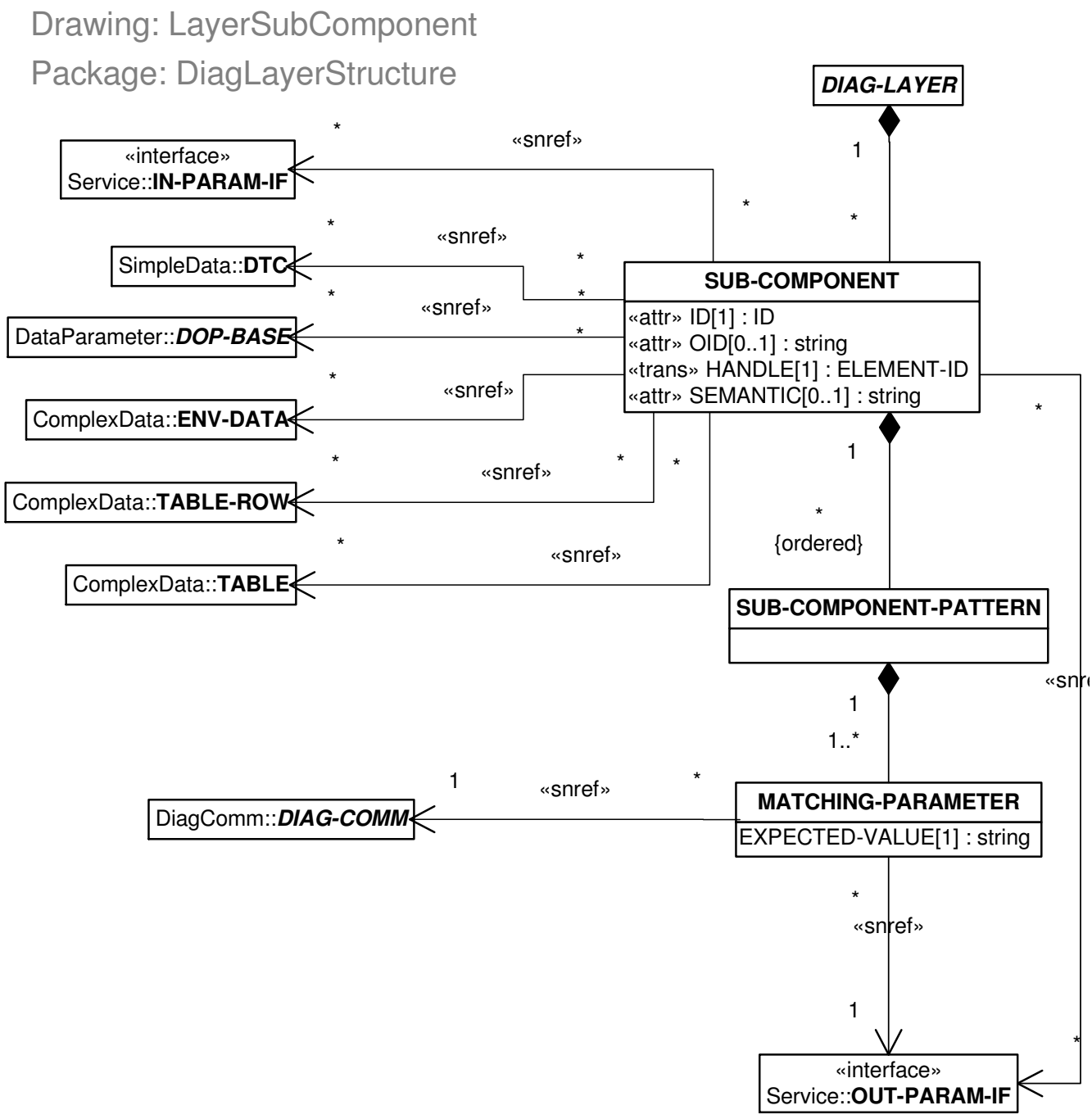


Figure 144 — SUBCOMPONENT

This element SUBCOMPONENT references all relevant ODX elements via ODX links

7.8.4 EXAMPLE (FUNCTION-DICTIONARY-SPEC)

```
<?xml version="1.0" encoding="UTF-8" ?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
- <FUNCTION-DICTIONARY-SPEC ID="X0001">
      <SHORT-NAME>FUNCDIC_Platform1</SHORT-NAME>
      <FUNCTION-NODES>
            <FUNCTION-NODE ID="FNC_Blinking-left">
                  <SHORT-NAME>FncBlinkingLeft</SHORT-NAME>
```

```
                    <LONG-NAME>Function blinking left</LONG-NAME>
                    <DESC><p>Periodic on/off lights left</p></DESC>
                    <VEHICLE-INFORMATION-CONNECTORS>
                          <VEHICLE-INFORMATION-CONNECTOR ID="Vhc_FrontEcu">
                                <SHORT-NAME>Vhc_FrontEcu</SHORT-NAME>
                                >LONG-NAME>All diagnostic elements on Front ECU
for function Blinking left</LONG-NAME>
                                <BASE-VARIANTS>
                                      <BASE-VARIANT ID-REF="EV_Front"/>
                                <BASE-VARIANTS>
                                <DIAG-OBJECT-CONNECTOR ID="dlc_1">
                                      <SHORT-NAME>sdfsdf</SHORT-NAME>
                                      <DTCS>
                                            <DTC-REF ID-REF="DTC_Pin2-
NoSignal"/>
                                            <DTC-REF ID-REF="DTC_Pin23-
NoSignal"/>
                          …
                                      </DTCS>
                                      <DIAG-COMMS>
                                            <DIAG-COMM-REF ID-
REF="DiagService_IOCPin_1"/>
                                            <DIAG-COMM-REF ID-
REF="DiagService_IOCPin_2"/>
                                      </DIAG-COMMS>
                                      <EXECUTABLE>
                                            <DIAG-COMM-REF ID-
                                      REF="SingleEcuJob_TestAllPins"/>
                                      <EXECUTABLE>
                                </DIAG-OBJECT-CONNECTOR>
                          </VEHICLE-INFORMATION-CONNECTOR>
                    </VEHICLE-INFORMATION-CONNECTORS>
              </FUNCTION-NODE>
       </FUNCTION-NODES>
       <FUNCTION-NODE-GROUPS>
              <FUNCTION-NODE-GROUP ID="FG_AllBlinking">
                    <SHORT-NAME>FG_AllBlinking</SHORT-NAME>
                    <LONG-NAME>All blinking functions</LONG-NAME>
                    <FUNCTION-NODE-REFS>
                          <FUNCTION-NODE-REF ID-REF="FncBlinkingLeft"/>
                    </FUNCTION-NODE-REFS>
              </FUNCTION-NODE-GROUP>
       </FUNCTION-NODE-GROUPS>
</ODX>
```

8 DATA MODEL IMPLEMENTATION IN XML

8.1 CLASSIFIER

8.1.1 CLASSES

8.1.1.1 DEFAULT MAPPING

8.1.1.1.1 Description

The default mapping for classes will result in a XML Schema complex type definition for each class, using the class name as complex type name. The elements based on such a type will also use the class name as element name.

8.1.1.1.2 UML example

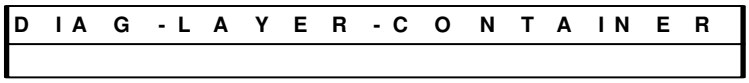


Figure 145 — Default class mapping

8.1.1.1.3 Schema example

```
<xsd:complexType name="DIAG-LAYER-CONTAINER">
     ...
</xsd:complexType>
...
<xsd:complexType name="ODX">
    <!--Class: ODX-->
    <xsd:choice>
        <xsd:element maxOccurs="1" minOccurs="1" type="DIAG-LAYER-CONTAINER"
        ...
    </xsd:choice>
    ...
</xsd:complexType>
```

8.1.1.1.4 ODX instance example

```
<ODX>
    <DIAG-LAYER-CONTAINER ID="ID0006">
    ...
    </DIAG-LAYER-CONTAINER>
</ODX>
```

8.1.1.2 TRANSPARENT MAPPING

8.1.1.2.1 Description

If a class has the stereotype «trans», it will be mapped to a named group definition. This makes such a class transparent (invisible) with respect to an ODX instance document.

8.1.1.2.2 UML example

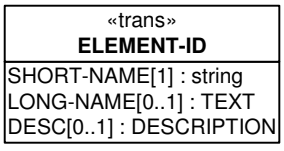


Figure 146 — Transparent class mapping

8.1.1.2.3 Schema example

```
<xsd:group name="ELEMENT-ID">
  <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string" name="SHORT-
NAME"/>
    <xsd:element maxOccurs="1" minOccurs="0" type="TEXT" name="LONG-NAME"/>
    <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION" name="DESC"/>
  </xsd:sequence>
</xsd:group>
```

8.1.1.2.4 ODX instance example

```
<some-surrounding-element>
  ...
  <SHORT-NAME>SOME_SHORT_NAME</SHORT-NAME>
  <LONG-NAME>SOME_LONG_NAME</LONG-NAME>
  ...
</some-surrounding-element>
```

8.1.1.3 INTERFACES

8.1.1.3.1 Description

Interfaces are identified by the «interface» stereotype. Interfaces can only be used as targets for short-name references or odx-links. The interface name determines the name of the linking element.

8.1.1.3.2 UML example

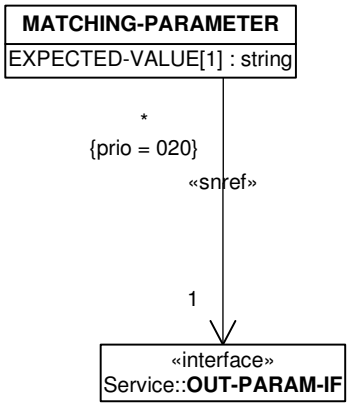


Figure 147 — Mapping interfaces

8.1.1.3.3 Schema example

```
<xsd:element maxOccurs="1" minOccurs="1" name="OUT-PARAM-IF-SNREF" type="SNREF"/>
```

8.1.1.3.4 ODX instance example

```
<OUT-PARAM-IF-SNREF SHORT-NAME="..."/>
```

8.1.1.4 ENUMERATIONS

8.1.1.4.1 Description

Enumerations are mapped to simple type definitions that are restrictions of the XML Schema built-in simple type *xsd:string*.

8.1.1.4.2 UML example

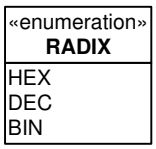


Figure 148 — Enumerations

8.1.1.4.3 Schema example

```
<xsd:simpleType name="RADIX">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="HEX"/>
        <xsd:enumeration value="DEC"/>
        <xsd:enumeration value="BIN"/>
    </xsd:restriction>
</xsd:simpleType>
...
<xsd:complexType name="PHYSICAL-TYPE">
  ...
  <xsd:attribute use="optional" type="RADIX" name="DISPLAY-RADIX"/>
</xsd:complexType>
```

8.1.1.4.4 ODX instance example

```
<PHYSICAL-TYPE DISPLAY-RADIX="DEC"/>
```

8.1.1.5 ODX-LINK CLASSES

8.1.1.5.1 Description

Classes with the stereotype «odxlink» can only occur in conjunction with an association having the same stereotype. This is semantically equivalent to association classes with «odxlink» stereotype and has been used to overcome a Visio deficiency that doesn't allow showing association classes on more than one page as well as the improper handling of generalisation. All association classes within the ODX model have to have the «odxlink» stereotype.

8.1.1.5.2 UML example

Figure 149 — Association class mapping

8.1.1.5.3 Schema example

```
<xsd:complexType name="COMPARAM-REF">
  <!--Association Class: COMPARAM-REF-->
  <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string" name="VALUE"/>
    <xsd:element maxOccurs="1" minOccurs="0" type="TEXT" name="DESC"/>
```

```
  </xsd:sequence>
  <xsd:attributeGroup ref="ODXLINK-ATTR"/>
</xsd:complexType>
...
<xsd:complexType name="FUNCTIONAL-GROUP-REF">
  <xsd:complexContent>
    <xsd:extension base="PARENT-REF"/>
  </xsd:complexContent>
</xsd:complexType>
...
<xsd:complexType name="PARENT-REF">
  <xsd:sequence>
    ...
  </xsd:sequence>
  <xsd:attributeGroup ref="ODXLINK-ATTR"/>
</xsd:complexType>
```

8.1.1.5.4 ODX instance example

```
<COMPARAM-REF DOCTYPE="CONTAINER" ID-REF="XID0001">
   <VALUE>0.125</VALUE>
</COMPARAM-REF>
...
<FUNCTIONAL-GROUP-REF DOCTYPE="CONTAINER" ID-REF="XID0003">
   <NOT-INHERITED-DIAG-COMMS>
     ...
   </NOT-INHERITED-DIAG-COMMS>
</ FUNCTIONAL-GROUP-REF>
```

8.1.1.6 IMPORTED CLASSES

8.1.1.6.1 Description

The stereotype «import» denotes classes that are imported at the XML Schema level and are therefore not described any further within the UML model. The class name will be usable as a type name within a XML Schema element definition.

8.1.1.6.2 UML example

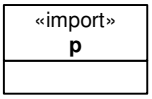


Figure 150 — Mapping of imported classes

8.1.1.6.3 Schema example

```
<xsd:element maxOccurs="unbounded" minOccurs="0" type="p" name="p"/>
```

8.1.1.6.4 ODX instance example

```
<some-surrounding-element>
  ...
  <p>Some text or other XHTML elements</p>
  ...
</some-surrounding-element>
```

8.1.2 ATTRIBUTES

8.1.2.1 DEFAULT MAPPING

8.1.2.1.1 Description

By default, UML attributes will be mapped to sub-elements of the corresponding class using the specified data type. The specified multiplicity will be transformed into the XML Schema occurrence constraints.

8.1.2.1.2 UML example

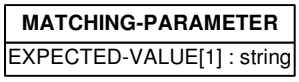


Figure 151 — Default attribute mapping

8.1.2.1.3 Schema example

```
<xsd:complexType name="MATCHING-PARAMETER">
  <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string" name="EXPECTED-
VALUE"/>
    ...
  </xsd:sequence>
</xsd:complexType>
```

8.1.2.1.4 ODX instance example

```
<MATCHING-PARAMETER>
  <EXPECTED-VALUE>SOME_VALUE</EXPECTED-VALUE>
</MATCHING-PARAMETER>
```

8.1.2.2 XML ATTRIBUTE MAPPING

8.1.2.2.1 Description

If an attribute has a stereotype <<attr>> it will be mapped to a XML attribute. The multiplicity is restricted to [0..1] or [1].
If the multiplicity is [0..1], a default value may be specified which will be mapped to a default constraint in XML. If, in addition to a default value, the attributes *changeability* is *frozen* it will be mapped to a fixed attribute.

8.1.2.2.2 UML example

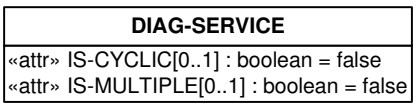


Figure 152 — XML attribute mapping

8.1.2.2.3 Schema example

```
<xsd:complexType name="DIAG-SERVICE">
  ...
  <xsd:attribute default="false" use="optional" type="xsd:boolean" name="IS-
CYCLIC"/>
  <xsd:attribute default="false" use="optional" type="xsd:boolean" name="IS-
MULTIPLE"/>
  ...
</xsd:complexType>
```

8.1.2.2.4 ODX instance example

```
<DIAG-SERVICE IS-CYCLIC=”true”>
  ...
</DIAG-SERVICE>
```

8.1.2.3 TRANSPARENT MAPPING

8.1.2.3.1 Description

If an attribute has the stereotype «trans», it will be mapped to a named group definition. This makes such an attribute transparent (invisible) with respect to an ODX instance document. It is important that the referenced class also has the «trans» stereotype.

8.1.2.3.2 UML example

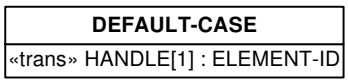


Figure 153 — Transparent attribute mapping

8.1.2.3.3 Schema example

```
<xsd:complexType name="DEFAULT-CASE">
  <xsd:sequence>
    <xsd:group ref="ELEMENT-ID"/>
    ...
  </xsd:sequence>
</xsd:complexType>
...
<xsd:group name="ELEMENT-ID">
  <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string" name="SHORT-
NAME"/>
    <xsd:element maxOccurs="1" minOccurs="0" type="TEXT" name="LONG-NAME"/>
    <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION" name="DESC"/>
  </xsd:sequence>
</xsd:group>
```

8.1.2.3.4 ODX instance example

```
<DEFAULT-CASE>
  <SHORT-NAME>SOME_SHORT_NAME</SHORT-NAME>
  <LONG-NAME>SOME_LONG_NAME</LONG-NAME>
</DEFAULT-CASE>
```

8.1.2.4 XML CONTENT MAPPING

8.1.2.4.1 Description

If an UML attribute has the stereotype «content», it will be mapped to the content of the class’ corresponding XML element. Not more than one attribute per class may have this stereotype. The multiplicity is restricted to [0..1] or [1].

8.1.2.4.2 UML example

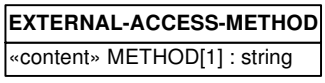


Figure 154 — XML content mapping

8.1.2.4.3 Schema example

```
<xsd:complexType name="EXTERNAL-ACCESS-METHOD">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string"/>
  </xsd:simpleContent>
</xsd:complexType>
```

8.1.2.4.4 ODX instance example

```
<EXTERNAL-ACCESS-METHOD>SOME_METHOD_NAME</EXTERNAL-ACCESS-METHOD>
```

8.2 RELATIONSHIPS

8.2.1 GENERALISATIONS

8.2.1.1 OVERVIEW

The ODX model uses two different implementation schemes of generalisation denoted by the «impl-parent» and «impl-child» stereotypes. The implementation scheme must not change in a single inheritance tree. The XML Schema types generated for both methods are identical however they are referenced differently.

8.2.1.2 PARENT GENERALISATION

8.2.1.2.1 Description

Using the *parent generalisation* concept, there are only XML elements created using the most general class of an inheritance tree. The actual derived type is being identified by the `xsi:type` attribute that is part of the XML Schema instance namespace.

8.2.1.2.2 UML example

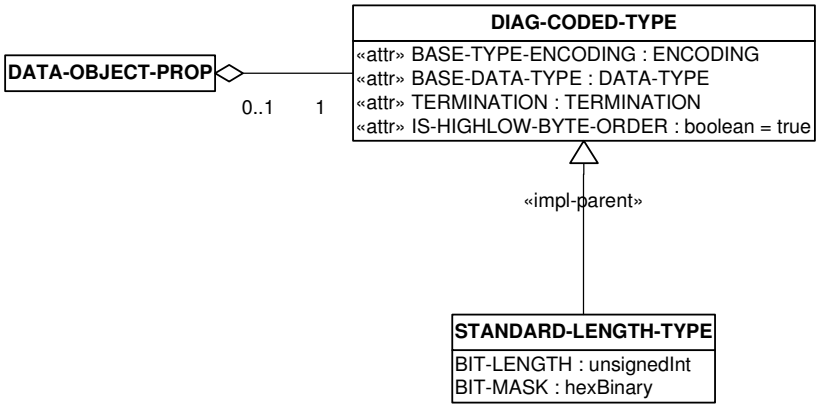


Figure 155 — Parent generalisation

8.2.1.2.3 Schema example

```
<xsd:complexType name="DIAG-CODED-TYPE">
  <!--Class: DIAG-CODED-TYPE-->
  <xsd:attribute use="optional" type="ENCODING" name="BASE-TYPE-ENCODING"/>
  <xsd:attribute use="required" type="DATA-TYPE" name="BASE-DATA-TYPE"/>
  <xsd:attribute use="optional" type="TERMINATION" name="TERMINATION"/>
  <xsd:attribute default="true" use="optional" type="xsd:boolean"
```

```
                name="IS-HIGHLOW-BYTE-ORDER"/>
</xsd:complexType>
...
<xsd:complexType name="STANDARD-LENGTH-TYPE">
  <xsd:complexContent>
  <xsd:extension base="DIAG-CODED-TYPE">
    <xsd:sequence>
      <xsd:element maxOccurs="1" minOccurs="1" type="xsd:unsignedInt" name="BIT-
LENGTH"/>
       <xsd:element maxOccurs="1" minOccurs="0" type="xsd:hexBinary" name="BIT-
MASK"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
...
<xsd:complexType name="DATA-OBJECT-PROP">
  <xsd:complexContent>
    <xsd:extension base="DOP-BASE">
      <xsd:sequence>
        ...
        <xsd:element maxOccurs="1" minOccurs="1" type="DIAG-CODED-TYPE"
                     name="DIAG-CODED-TYPE"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

8.2.1.2.4 ODX instance example

```
<DATA-OBJECT-PROP>
  <SHORT-NAME>DTC_HEX_VALUE</SHORT-NAME>
  ...
  <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32" BASE-TYPE-ENCODING="2C"
                   xsi:type="STANDARD-LENGTH-TYPE">
    <BIT-LENGTH>8</BIT-LENGTH>
  </DIAG-CODED-TYPE>
  ...
</DATA-OBJECT-PROP>
```

8.2.1.3 CHILD GENERALISATION

8.2.1.3.1 Description

Using the *child generalisation* concept, there are only XML elements created for the most specialised classes of an inheritance tree.

8.2.1.3.2 UML example

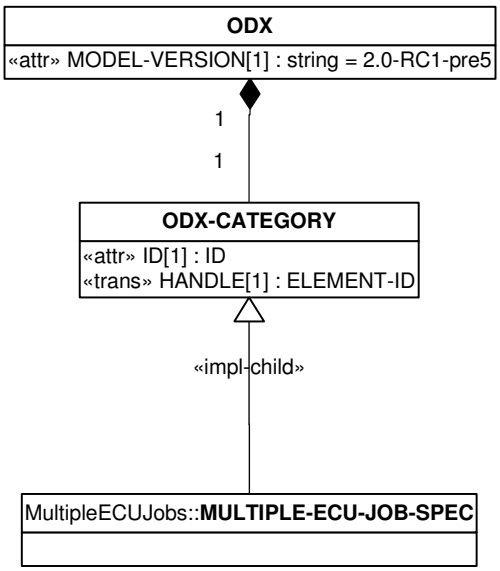


Figure 156 — UML example of child generalisation

8.2.1.3.3 Schema example

```
<xsd:complexType name="ODX-CATEGORY">
  ...
</xsd:complexType>
...
<xsd:complexType name="MULTIPLE-ECU-JOB-SPEC">
  <xsd:complexContent>
    <xsd:extension base="ODX-CATEGORY">
      ...
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
...
<xsd:complexType name="ODX">
  <xsd:choice>
    ...
    <xsd:element maxOccurs="1" minOccurs="1" type="MULTIPLE-ECU-JOB-SPEC"
                 name="MULTIPLE-ECU-JOB-SPEC"/>
    ...
  </xsd:choice>
  ...
</xsd:complexType>
```

8.2.1.3.4 ODX instance example

```
<ODX>
  < MULTIPLE-ECU-JOB-SPEC>
  ...
  </ MULTIPLE-ECU-JOB-SPEC>
</ODX>
```

8.2.2 ASSOCIATIONS

8.2.2.1 COMPOSITION AND AGGREGATION

8.2.2.1.1 Description

Composition and aggregation are both mapped to identical XML structures. This is done in a way that the aggregated object becomes a sub-element of the aggregating object. The multiplicities given at the association end of the aggregated class are mapped to XML Schema occurrence constraints. The multiplicities given at the other end are irrelevant for the implementation.
If the upper bound of the specified multiplicity exceeds 1, the association will be mapped to a wrapper element as described in the following section.

8.2.2.1.2 UML example

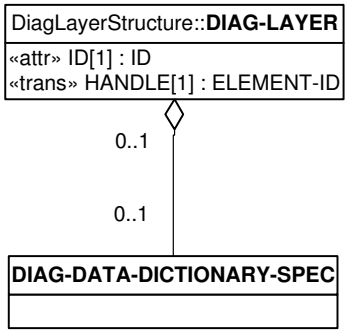


Figure 157 — UML example of composition and aggregation

8.2.2.1.3 XML Schema example

```
<xsd:complexType name="DIAG-LAYER">
  <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="0" type="DIAG-DATA-DICTIONARY-SPEC"
                 name="DIAG-DATA-DICTIONARY-SPEC"/>
  </xsd:sequence>
  ...
</xsd:complexType>
```

8.2.2.1.4 ODX instance example

```
<PROTOCOL>
  ...
  <DIAG-DATA-DICTIONARY-SPEC>
    ...
  </DIAG-DATA-DICTIONARY-SPEC>
  ...
<PROTOCOL>
```

8.2.2.2 WRAPPER ELEMENTS

8.2.2.2.1 Description

If the upper bound of the specified multiplicity of an association exceeds 1, the association will be mapped to either one or multiple wrapper elements, which are implemented by anonymous type definition. The specific implementation of the wrapper element depends on the generalisation concept (see 8.2) being used for the aggregated class:

8.2.2.2.2 Wrapper element for child generalisations

8.2.2.2.2.1.1 Description

Due to the fact that wrapper elements have been introduced to group elements of identical type, it requires multiple wrapper elements to group the distinct elements resulting from child generalisation. I.e. there will be one wrapper element for each most specialised class within an inheritance tree using child generalisation.

UML example

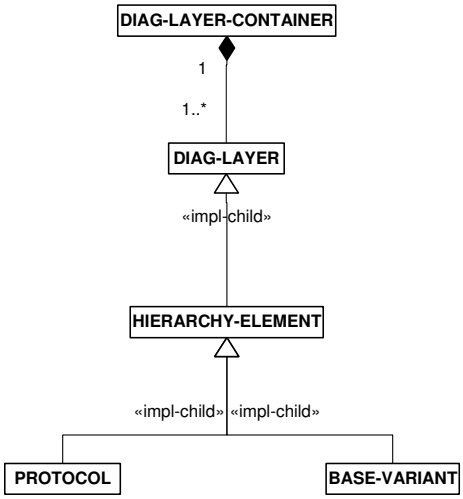


Figure 158 — Wrapper for child generalisations

XML Schema example

```
<xsd:complexType name="DIAG-LAYER-CONTAINER">
  <xsd:complexContent>
    <xsd:extension base="ODX-CATEGORY">
      <xsd:sequence>
        <xsd:element maxOccurs="1" minOccurs="0" name="PROTOCOLS">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" minOccurs="1" type="PROTOCOL"
                           name="PROTOCOL"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        ...
        <xsd:element maxOccurs="1" minOccurs="0" name="BASE-VARIANTS">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" minOccurs="1" type="BASE-
VARIANT"
                           name="BASE-VARIANT"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        ...
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

ODX instance example

```
<DIAG-LAYER-CONTAINER>
  <PROTOCOLS>
```

```
    <PROTOCOL ID="ID04Mar26100629">
      ...
    </PROTOCOL>
  </PROTOCOLS>
  <BASE-VARIANTS>
    <BASE-VARIANT ID="ID04Mar26100633">
      ...
    </BASE-VARIANT>
  </BASE-VARIANTS>
</DIAG-LAYER-CONTAINER>
```

8.2.2.2.3 Wrapper element for parent generalisations

Description

As stated above, wrapper elements have been introduced to group elements of identical type. In case of the parent generalisation there exist only elements of the most general type. Therefore a single wrapper element for all specialisations is sufficient.

UML example

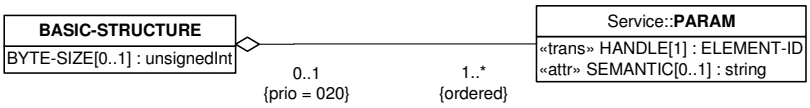


Figure 159 — Wrapper for parent generalisations

XML Schema example

```
<xsd:complexType name="BASIC-STRUCTURE">
  ...
  <xsd:sequence>
    ...
    <xsd:element maxOccurs="1" minOccurs="1" name="PARAMS">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="1" type="PARAM"
name="PARAM"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    ...
  </xsd:sequence>
  ...
</xsd:complexType>
```

ODX instance example

```
<STRUCTURE>
  <PARAMS>
    <PARAM xsi:type="VALUE"/>
  </PARAMS>
</STRUCTURE>
```

8.2.2.2.4 No wrapper

Description

There are some cases where the presence of a wrapper element is not desirable. This is mostly due to compatibility reasons with other ASAM standards. This circumstance is being denoted by the **«nowrapper»** constraint.

UML example

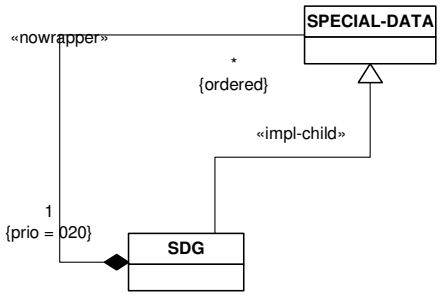


Figure 160 — No wrapper element

XML Schema example

```
<xsd:complexType name="SDG">
  <xsd:complexContent>
    <xsd:extension base="SPECIAL-DATA">
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" minOccurs="0" type="SDG" name="SDG"/>
        ...
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

ODX instance example

```
<SDG>
  <SDG/>
  <SDG/>
</SDG>
```

8.2.2.2.5 Single wrapper for different elements

Description

In several other cases there was a requirement to collect different elements within a single wrapper element that is being shown by a constraint attached to all affected associations. The body of the constraint contains also the name of the single wrapper element and the total minimum multiplicity. The common-wrapper constraint takes precedence over the default wrapper definition.

The rules for name generation apply as usual, e.g. -REFS will be appended to the name of the wrapper element. All associations that participate in a common wrapper constraint must have the same *prio*-number.

The common-wrapper constraint works only for identical associations.

UML example

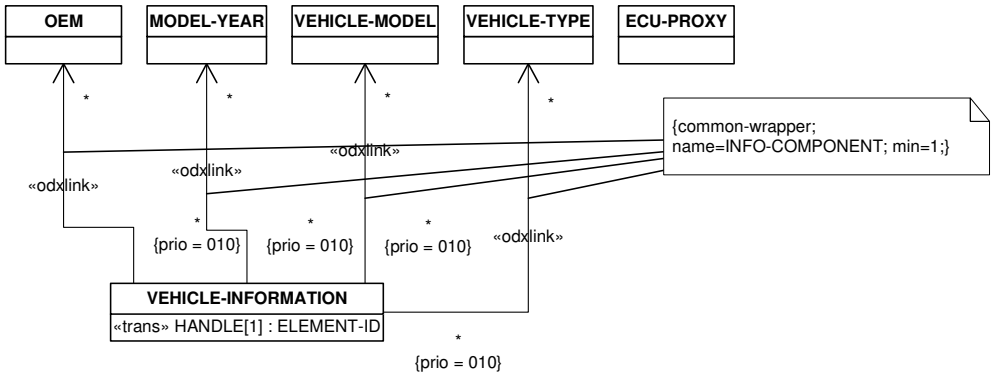


Figure 161 — Common wrapper element

XML Schema example

```
<xsd:complexType name="VEHICLE-INFORMATION">
  <xsd:sequence>
    ...
    <xsd:element maxOccurs="1" minOccurs="1" name="INFO-COMPONENT-REFS">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="1" name="INFO-COMPONENT-
REF"
                       type="ODXLINK"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    ...
  </xsd:sequence>
</xsd:complexType>
```

ODX instance example

```
<VEHICLE-INFORMATION>
  <INFO-COMPONENT-REFS>
    <INFO-COMPONENT-REF ID-REF="ID001"/>
    <INFO-COMPONENT-REF ID-REF="ID002"/>
  </INFO-COMPONENT-REFS>
</VEHICLE-INFORMATION>
```

9 ODX PACKAGING (PDX)

9.1 OVERVIEW

The exchange of ODX data can be realized with a file container called "PDX package". All files to be exchanged are packaged in one PDX package. A PDX package can handle ODX-files of different types with the following file extensions:

- odx-c (for COMPARAM-SPEC)
- odx-d (for DIAG-LAYER-CONTAINER)
- odx-e (for ECU-CONFIG)

- odx-f (for FLASH)
- odx-fd (for FUNCTION-DICTIONARY)
- odx-m (for MULTIPLE-ECU-JOB-SPEC)
- odx-v (for VEHICLE-INFORMATION-SPEC)
- odx (alternatively all the files containing ODX data can use the extension "odx" lesser restrictive.)

The PDX-package may not only contain ODX data, text or pictures, but arbitrary files of just any format.
The content of a PDX package is described in a separate XML document called "PDX package catalogue". The PDX package catalogue itself contains no ODX data. It is used as a repository for meta-data in the exchange process.
All file names for files containing ODX data or involved in the ODX/PDX exchange proccess have to be handled in a case-sensitive manner by all ODX-compliant tools. The file-extensions shall always be expressed in small letters. The filename of the PDX package catalogue shall always be "index.xml" in small letters.

PDX package/package catalogue use cases

- Mutual data exchange between OEM and partners:
  - Full data exchange (one way or two ways)
  - Incremental (partial) data exchange: only newly created or changed data is exchanged
- Configuration management and version control:
- Exchange of revision history
- Support for CMS[15)] -specific versioning: each process partner can add CMS-internal revision information independently
- Multi-versioning: different revisions of a file may be provided in a single PDX package

9.2 STRUCTURE OF PDX PACKAGE

9.2.1 STRUCTURE OF PDX PACKAGE CATALOGUE

A PDX package consists of a collection of files. The content of the PDX package is described in the PDX package catalogue, which is one of the files that are stored in the PDX package.

15) Configuration Management System

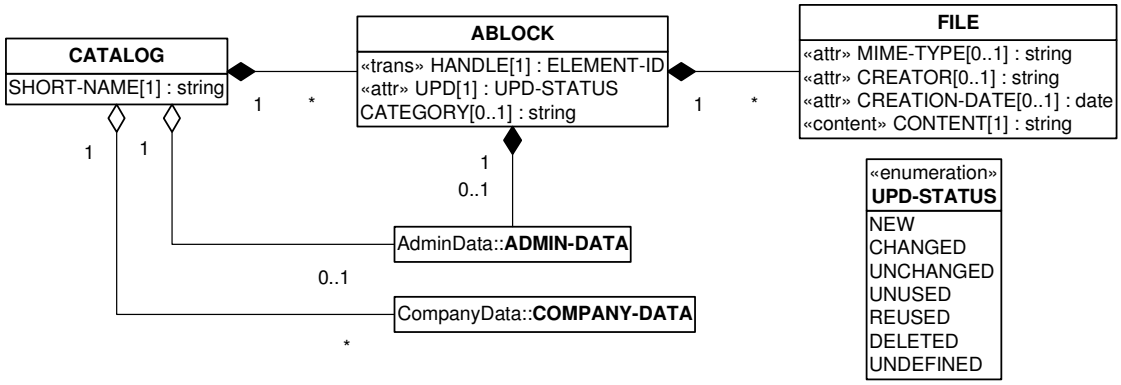


Figure 162 — Structure of the PDX package catalogue

A PDX package catalogue is represented by a CATALOG instance; it provides a list of all files that are stored in the corresponding PDX package. These file entries can be grouped logically. As an example, ODX data files may be stored in one group, and files for documentation (manuals, images, etc.) in another one.
For this purpose the CATALOG contains a list of so-called ABLOCKs; each ABLOCK represents a logical group and may contain an arbitrary number of FILE members each representing a single file in the PDX package. Optionally, a CATEGORY member may be specified for an ABLOCK. This allows the definition of process-specific criteria for the categorisation of files. The following values are predefined for CATEGORY:

a. ODX-DATA for all files that are derived from an ODX-CATEGORY

b. ODX-JOB for all job code files (e.g. java-files, class-files, jar-files, dll-files)

c. LIB for all libraries that can be used or are imported by jobs

d. PROGRAMMING-DATA for binary/hex files that are used for ECU programming and are referenced from an ECU-MEM

The SHORT-NAME is a mandatory member of each ABLOCK. It identifies the logical group that the files in the ABLOCK belong to rather than the ABLOCK itself. Multiple ABLOCKs with the same SHORT-NAME may exist in one PDX package catalogue.
Each ABLOCK may contain an optional ADMIN-DATA member to store revision information that is specific for the ABLOCK, i.e. it refers to all FILEs in the ABLOCK. ADMIN-DATA that is valid for all files in the PDX package and other global meta-data like COMPANY-DATA can be stored directly in the CATALOG (see section 7.1.2.3 for a detailed description of ADMIN-DATA and COMPANY-DATA).
An ABLOCK has a mandatory UPD member that describes the update status of the ABLOCK's files within the data exchange process. The UPD attribute can have one of the following values:

1. NEW New files have been introduced.

2. CHANGED The files have been changed.

3. UNCHANGED The files have not been changed.

    a. UNUSED The files are no longer used at the moment but may be used again in the future.

    b. REUSED The files that have been marked as UNUSED in an earlier stage of the data exchange process are used again.

c. DELETED The files have been deleted.

d. UNDEFINED No update status available. This is the default value.

Each FILE member specifies the file name to locate one file within the PDX package. The MIME-TYPE, the CREATOR, and the CREATION-DATE of such a file can be specified with optional attributes. As values of MIME-TYPE the already standardized MIME types shall be used. Since ODX documents are XML documents the MIME-TYPE "text/xml" should be used for them. The following table defines some MIME-TYPEs, which should be used in catalogues:

Table 14 — MIME-TYPEs used in PDX packages

File extension	MIME-TYPE
.class	application/x-java-vm
.java	text/plain
.jar	application/x-java-archive
.odx (odx-f, odx-c, …)	text/xml
.xml	text/xml
.pdf	application/pdf
.hex (flash data)	application/octet-stream
.zip	application/zip

EXAMPLE A catalogue for a PDX package with three files

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG xsi:noNamespaceSchemaLocation="odx-cc.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
       <SHORT-NAME>ODX_CC_Example</SHORT-NAME>
       <ABLOCKS>
              <ABLOCK UPD="NEW">
                     <SHORT-NAME>DiagData</SHORT-NAME>
                     <CATEGORY>ODX-DATA</CATEGORY>
                     <FILES>
                            <FILE MIME-TYPE="text/xml" CREATOR="xyz" CREATION-
DATE="2005-12-08">DiagData.odx</FILE>
                     </FILES>
              </ABLOCK>
              <ABLOCK UPD="NEW">
                     <SHORT-NAME>Documentation</SHORT-NAME>
                     <CATEGORY>DOCUMENTATION</CATEGORY>
                     <FILES>
                            <FILE MIME-TYPE="application/pdf" CREATOR="xyz"
CREATION-DATE="2005-12-08">odx_spec.pdf</FILE>
                            <FILE MIME-TYPE="text/html" CREATOR="xyz" CREATION-
DATE="2005-12-08">readme.html</FILE>
                     </FILES>
              </ABLOCK>
       </ABLOCKS>
</CATALOG>
```

9.2.2 TECHNICAL ASPECTS OF PDX PACKAGE

A PDX package is implemented as a ZIP archive with the file extension “.pdx” (for “packaged ODX”).
A PDX package contains one file with the PDX package catalogue (see example above). This file is named "index.xml" and is located in the root directory of the PDX package’s internal file system hierarchy.

A PDX package is a self-contained file, i.e. the PDX package catalogue references only files within the PDX package. As a consequence, no references to external files are allowed in the PDX package catalogue.
A PDX package supports the storage of files in a hierarchical directory structure. In case of hierarchical file storage, all file paths in the PDX package catalogue are formulated as absolute paths starting with the root directory of the PDX package's internal file system hierarchy (e.g. “zipdirectory1/zipdirectory2/myfile.odx”). In case of java jobs the full path resulting from the java package structure needs to be included in the PDX file to preserve the unambiguousness of a java job. Naming conventions for files in PDX packages.

A file name is usually built according to the following scheme:
NAME.EXTENSION
The type of a file is specified explicitly by the MIME-TYPE member. As a consequence, there's no need for tools conforming to ODX to map the file extension to a MIME type. The file extension should be handled as a part of the file name.
To store multiple revisions of the same file within a single ODX container (which is called "multi-versioning" in ODX) each file revision is stored in a separate ABLOCK. All ABLOCKs that contain revisions of the same file share the same SHORT-NAME. Different revisions of a file must not be stored with the same file name in the PDX package. Therefore, each revision of the file is marked with an identifier that is appended to the file name. This will result in the following extended scheme for file names with revision information:
NAME.EXTENSION.REVISION
The structure of the revision identifier is process-specific. The creator of the PDX package just has to take care that the revision identifiers are unique for all revisions of the file. Furthermore, revision identifiers should not contain any dots (".") to simplify the splitting of the file name and the revision identifier. As an example, the revision "1.2.3" of the file "DiagLayers.odx" can be stored with the name "DiagLayers.odx.123"
If no multi-versioning is applied then the revision identifiers can be safely omitted from the file name.
Stripping of revision identifiers

A CMS typically expects that all revisions of a file have the same file name. As a consequence, any revision identifier must be stripped from the file name before the revision is checked in to the CMS.
The tool that unpacks the PDX package has to determine whether it has to strip the revision identifier from the file name: the existence of two or more ABLOCKs with the same SHORT-NAME indicates multi-versioning, i.e. the file name has a revision identifier appended.

9.3 USAGE SCENARIOS

9.3.1 USAGE OF PDX PACKAGE IN THE EXCHANGE PROCESS

The exchange of ODX data and other supporting files is the main purpose of an PDX package.
A process partner builds the PDX package by including all files he wants to transfer and by describing those in the PDX package catalogue (see figure below). The other process partner receives the PDX package and extracts the files; the PDX package catalogue can provide information how these files have to be handled according to the data exchange process.

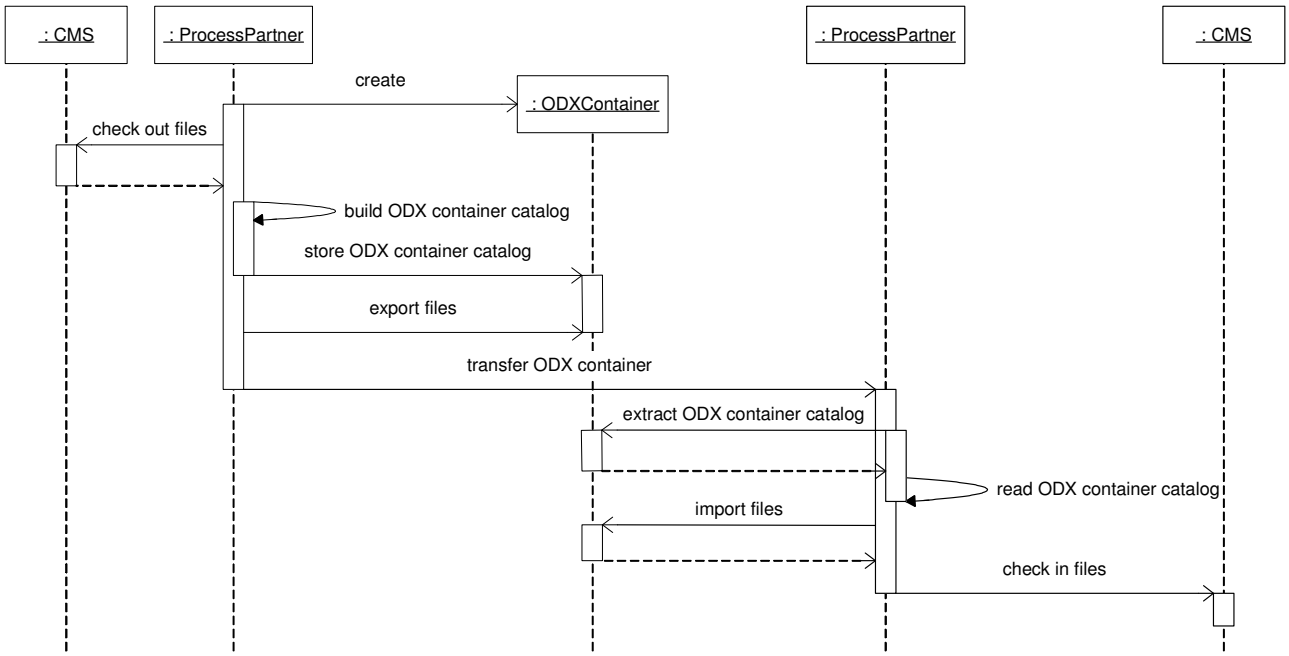


Figure 163 — Usage of an ODX container in the data exchange process

Full data exchange

All diagnostic layer instances that are needed for the diagnosis are packaged in one PDX package. Supporting files, e.g. manuals or images, are added to the PDX package.

Incremental (partial) data exchange

To reduce the amount of transferred data only new or changed files are included in the PDX package. All unchanged files are omitted from the PDX package; they are listed in the PDX package catalogue using an ABLOCK with the UPD member set to "UNCHANGED". Unused/reused files can be handled the same way by setting UPD to UNUSED/REUSED.

The process partners have to agree on a common set of files as a base for the data exchange process. Such a common set may be established by a full data exchange at the beginning of the data exchange process.

NOTE The process partners are free to decide whether they retain ABLOCKs in the PDX package catalogue that were not changed since the last data exchange. Including all unchanged ABLOCKs in the PDX package catalogue finally results in a list of all files involved in the data exchange process so far. Such a list is useful for the documentation of the data exchange process. However, the PDX package catalogue may become very large and difficult to maintain.

9.3.2 CONFIGURATION MANAGEMENT AND VERSION CONTROL

ODX is designed to be independent from the used tools and the applied process to support the exchangeability of ODX data. As a consequence, process-specific information like storage structure or file revisions should be kept outside of the ODX data. The PDX package catalogue can be used to transfer this kind of information from one process partner to the other.

Generally, we distinguish two types of revision information: Public revision information can be used by all process partners, while internal revision information is provided for company-internal usage only (see section 7.1.2.3 for details).

Exchange of revision history

In some cases the complete public revision history of a file has to be exchanged, e.g. if a process partner gets a file for the first time that was changed several times before.
A public revision history of a file is created by adding a new DOC-REVISION member to the ADMIN-DATA of the file's ABLOCK for every new revision of the file. If an ABLOCK contains more than one file then all the files have the same revision history.
To preserve the public revision history of a file the process partners have to retain the file's ABLOCK and its associated ADMIN-DATA until the next file revision is exchanged. Otherwise any previously accumulated revision information would be lost.
EXAMPLE Exchange of revision history

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG xsi:noNamespaceSchemaLocation="odx-cc.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
	<SHORT-NAME>ODX_CC_DR_Example</SHORT-NAME>
	<COMPANY-DATAS>
		<COMPANY-DATA ID="abc">
			<SHORT-NAME>abc</SHORT-NAME>
			<LONG-NAME>ABC Corp.</LONG-NAME>
		</COMPANY-DATA>
		<COMPANY-DATA ID="xyz">
			<SHORT-NAME>xyz</SHORT-NAME>
			<LONG-NAME>XYZ Inc.</LONG-NAME>
			<TEAM-MEMBERS>
				<TEAM-MEMBER ID="jsmith">
					<SHORT-NAME>jsmith</SHORT-NAME>
					<LONG-NAME>John Smith</LONG-NAME>
				</TEAM-MEMBER>
			</TEAM-MEMBERS>
		</COMPANY-DATA>
	</COMPANY-DATAS>
	<ABLOCKS>
		<ABLOCK UPD="CHANGED">
			<SHORT-NAME>DiagData</SHORT-NAME>
			<CATEGORY>ODX-DATA</CATEGORY>
			<ADMIN-DATA>
				<LANGUAGE>en_UK</LANGUAGE>
				<DOC-REVISIONS>
					<DOC-REVISION>
						<REVISION-LABEL>1.0.0</REVISION-LABEL>
						<STATE>draft</STATE>
						<DATE>2005-12-08T13:55:13</DATE>
					</DOC-REVISION>
					<DOC-REVISION>
						<TEAM-MEMBER-REF ID-REF="jsmith"/>
						<REVISION-LABEL>1.0.1</REVISION-LABEL>
						<STATE>release</STATE>
						<DATE>2005-12-08T15:55:14</DATE>
						<MODIFICATIONS>
							<MODIFICATION>
								<CHANGE>added diagnostic
service</CHANGE>
							</MODIFICATION>
						</MODIFICATIONS>
					</DOC-REVISION>
				</DOC-REVISIONS>
			</ADMIN-DATA>
			<FILES>
				<FILE MIME-TYPE="text/xml" CREATOR="xyz" CREATION-
DATE="2005-12-08">DiagData.odx</FILE>
			</FILES>
		</ABLOCK>
	</ABLOCKS>
</CATALOG>
```

CMS-specific versioning

If a process partner manages ODX files with a CMS then this CMS typically applies its own mechanism for version control. This includes an own versioning scheme etc. CMS-specific data is stored as internal revision information within the PDX package catalogue to allow accurate version tracking independently from the public revision information. Whenever the CMS of a process partner updates its internal revision information of a file a new COMPANY-REVISION-INFO member is added to the latest DOC-REVISION of the file. The COMPANY-REF in COMPANY-REVISION-INFO is used to identify the process partner who created that internal revision information. This allows the process partners to store their CMS-specific revision histories in the PDX package catalogue without interfering with each other.
If CMS-specific revision information is needed then a separate ABLOCK should be provided for each file within the PDX package. This way, process partners can update internal revision information independently. As with the public revision history the process partners have to retain the ADMIN-DATA.
EXAMPLE CMS-specific versioning

EXAMPLE CMS-specific versioning

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG xsi:noNamespaceSchemaLocation="odx-cc.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
       <SHORT-NAME>ODX_CC_CRI_Example</SHORT-NAME>
       <COMPANY-DATAS>
              <COMPANY-DATA ID="abc">
                     <SHORT-NAME>abc</SHORT-NAME>
                     <LONG-NAME>ABC Corp.</LONG-NAME>
              </COMPANY-DATA>
              <COMPANY-DATA ID="xyz">
                     <SHORT-NAME>xyz</SHORT-NAME>
                     <LONG-NAME>XYZ Inc.</LONG-NAME>
              </COMPANY-DATA>
       </COMPANY-DATAS>
       <ABLOCKS>
              <ABLOCK UPD="CHANGED">
                     <SHORT-NAME>DiagData</SHORT-NAME>
                     <CATEGORY>ODX-DATA</CATEGORY>
                     <ADMIN-DATA>
                            <LANGUAGE>en_US</LANGUAGE>
                            <DOC-REVISIONS>
                                   <DOC-REVISION>
                                          <REVISION-LABEL>1.0.0</REVISION-LABEL>
                                          <STATE>draft</STATE>
                                          <DATE>2005-12-08T15:55:13</DATE>
                                          <COMPANY-REVISION-INFOS>
                                                 <COMPANY-REVISION-INFO>
                                                        <COMPANY-DATA-REF ID-
REF="xyz"/>
                                                        <REVISION-
LABEL>01_00</REVISION-LABEL>
                                                 </COMPANY-REVISION-INFO>
                                          </COMPANY-REVISION-INFOS>
                                   </DOC-REVISION>
                                   <DOC-REVISION>
                                          <REVISION-LABEL>1.0.1</REVISION-LABEL>
                                          <STATE>release</STATE>
                                          <DATE>2005-12-08T15:55:14</DATE>
                                          <COMPANY-REVISION-INFOS>
                                                 <COMPANY-REVISION-INFO>
                                                        <COMPANY-DATA-REF ID-
REF="xyz"/>
```

```
                                                        <REVISION-
LABEL>01_05</REVISION-LABEL>
                                                    </COMPANY-REVISION-INFO>
                                                </COMPANY-REVISION-INFOS>
                                            </DOC-REVISION>
                                        </DOC-REVISIONS>
                                </ADMIN-DATA>
                                <FILES>
                                        <FILE MIME-TYPE="text/xml" CREATOR="xyz" CREATION-
DATE="2005-12-08">DiagData.odx</FILE>
                                </FILES>
                        </ABLOCK>
                </ABLOCKS>
</CATALOG>
```

Using the UPD attribute for configuration management

During the export of a file to a PDX package the container creator has to set the UPD attribute of the ABLOCK that contains the file in the PDX package catalogue. To determine the correct update status of a file the container creator compares the current revision of the file to the last revision that was exchanged with the process partner. The revision information itself can be omitted from the file's ABLOCK if it's not needed for other purposes (e.g. for the revision history).
In an incremental data exchange process, the attribute values UNCHANGED, UNUSED, REUSED, and DELETED imply that the files listed in this ABLOCK don't have to be exported to the PDX package because the current revisions of the files were transferred during a previous data exchange. The container creator can include the files as a reference for the receiver of the PDX package.

- The receiver of a PDX package may use the UPD member of an ABLOCK as a hint how to import the files in the ABLOCK (e.g. to a CMS). This is useful if the PDX package catalogue provides no other revision information.

The import of a file could results in a versioning conflict if the receiver of the PDX package has changed this file since the last data exchange. Such a conflict has to be resolved manually by the process partners.

EXAMPLE Usage of UPD attribute in a complex data exchange process

The process partner "ABC" creates an PDX package with the revision 1.0.0 of the file "DiagData.odx" an the following PDX package catalogue:

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG xsi:noNamespaceSchemaLocation="odx-cc.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <SHORT-NAME>ODX_CC_UPD_Example_1</SHORT-NAME>
        <ABLOCKS>
                <ABLOCK UPD="NEW">
                        <SHORT-NAME>DiagData</SHORT-NAME>
                        <CATEGORY>ODX-DATA</CATEGORY>
                        <ADMIN-DATA>
                                <DOC-REVISIONS>
                                        <DOC-REVISION>
                                                <REVISION-LABEL>1.0.0</REVISION-LABEL>
                                                <DATE>2005-12-08T15:55:14</DATE>
                                        </DOC-REVISION>
                                </DOC-REVISIONS>
                        </ADMIN-DATA>
                        <FILES>
                                <FILE MIME-TYPE="text/xml" CREATOR="abc" CREATION-
DATE="2005-12-08">DiagData.odx</FILE>
                        </FILES>
```

```
            </ABLOCK>
      </ABLOCKS>
</CATALOG>
```

The process partner "XYZ" imports "DiagData.odx" from the received PDX package. He changes this file and exports it as revision 1.0.1 to the PDX package. He also adds two files "odx_spec.pdf" and "readme.html" without further revision information. The resulting PDX package catalogue may look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG xsi:noNamespaceSchemaLocation="odx-cc.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <SHORT-NAME>ODX_CC_UPD_Example_2</SHORT-NAME>
      <ABLOCKS>
            <ABLOCK UPD="CHANGED">
                  <SHORT-NAME>DiagData</SHORT-NAME>
                  <CATEGORY>ODX-DATA</CATEGORY>
                  <ADMIN-DATA>
                        <DOC-REVISIONS>
                              <DOC-REVISION>
                                    <REVISION-LABEL>1.0.0</REVISION-LABEL>
                                    <DATE>2005-12-08T15:55:14</DATE>
                              </DOC-REVISION>
                              <DOC-REVISION>
                                    <REVISION-LABEL>1.0.1</REVISION-LABEL>
                                    <DATE>2005-12-08T17:25:04</DATE>
                              </DOC-REVISION>
                        </DOC-REVISIONS>
                  </ADMIN-DATA>
                  <FILES>
                        <FILE MIME-TYPE="text/xml" CREATOR="xyz"
                   CREATION-DATE="2005-12-08">DiagData.odx</FILE>
                  </FILES>
            </ABLOCK>
            <ABLOCK UPD="NEW">
                  <SHORT-NAME>Documentation</SHORT-NAME>
                  <ADMIN-DATA>
                        <DOC-REVISIONS>
                              <DOC-REVISION>
                                    <REVISION-LABEL>1.0.0</REVISION-LABEL>
                                    <DATE>2005-12-08T17:25:04</DATE>
                              </DOC-REVISION>
                        </DOC-REVISIONS>
                  </ADMIN-DATA>
                  <FILES>
                        <FILE MIME-TYPE="application/pdf" CREATOR="xyz"
                   CREATION-DATE="2005-12-08">odx_spec.pdf</FILE>
                        <FILE MIME-TYPE="text/html" CREATOR="xyz"
                   CREATION-DATE="2005-12-08">readme.html</FILE>
                  </FILES>
            </ABLOCK>
      </ABLOCKS>
</CATALOG>
```

Checking the UPD attribute "ABC" notices that "DiagData.odx" was changed; he saves the revision 1.0.0 of this file before he imports the new revision and the other new files from the PDX package.
After changing the file "readme.html" "ABC" stores the new revision of this file in a PDX package; the other files haven't changed and are omitted from the PDX package. The content of the PDX package is described as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG xsi:noNamespaceSchemaLocation="odx-cc.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <SHORT-NAME>ODX_CC_UPD_Example_3</SHORT-NAME>
      <ABLOCKS>
```

```
            <ABLOCK UPD="UNCHANGED">
                  <SHORT-NAME>DiagData</SHORT-NAME>
                  <CATEGORY>ODX-DATA</CATEGORY>
                  <ADMIN-DATA>
                        <DOC-REVISIONS>
                              <DOC-REVISION>
                                    <REVISION-LABEL>1.0.0</REVISION-LABEL>
                                    <DATE>2005-12-08T15:55:14</DATE>
                              </DOC-REVISION>
                              <DOC-REVISION>
                                    <REVISION-LABEL>1.0.1</REVISION-LABEL>
                                    <DATE>2005-12-08T17:25:04</DATE>
                              </DOC-REVISION>
                        </DOC-REVISIONS>
                  </ADMIN-DATA>
                  <FILES>
                        <FILE MIME-TYPE="text/odx" CREATOR="xyz" CREATION-
DATE="2005-12-08">DiagData.odx</FILE>
                  </FILES>
            </ABLOCK>
            <ABLOCK UPD="CHANGED">
                  <SHORT-NAME>Documentation</SHORT-NAME>
                  <ADMIN-DATA>
                        <DOC-REVISIONS>
                              <DOC-REVISION>
                                    <REVISION-LABEL>1.0.0</REVISION-LABEL>
                                    <DATE>2005-12-08T17:25:04</DATE>
                              </DOC-REVISION>
                              <DOC-REVISION>
                                    <REVISION-LABEL>1.0.1</REVISION-LABEL>
                                    <DATE>2005-12-08T19:56:59</DATE>
                                    <MODIFICATIONS>
                                          <MODIFICATION>
                                                <CHANGE>readme.html was
revised</CHANGE>
                                          </MODIFICATION>
                                    </MODIFICATIONS>
                              </DOC-REVISION>
                        </DOC-REVISIONS>
                  </ADMIN-DATA>
                  <FILES>
                        <FILE MIME-TYPE="application/pdf" CREATOR="xyz"
CREATION-DATE="2005-12-08">odx_spec.pdf</FILE>
                        <FILE MIME-TYPE="text/html" CREATOR="abc" CREATION-
DATE="2005-12-08">readme.html</FILE>
                  </FILES>
            </ABLOCK>
      </ABLOCKS>
</CATALOG>
```

Annex A
(normative)

Enumerations and pre-defined values

A.1 Predefined values of SEMANTICs

Table A.1 — SEMANTIC at DIAG-COMM

Pre-defined value	Description
COMMUNICATION	To show that communication between tester and ECU is meant (Used with DIAGNOSTIC-CLASS=”STARTCOM” and DIAGNOSTIC-CLASS=”STOPCOM”)
SESSION	Switch a diagnostic session
SECURITY	Enable security access
IDENTIFICATION	Read ECU or flash identification
FAULTREAD	Read diagnostic trouble codes
FAULTCLEAR	Clear diagnostic trouble codes
DEFAULT-FAULT-CLEAR	Clear diagnostic trouble codes (may only appear once)
ENVREAD	Read environment data of diagnostic trouble codes
VARCODING	Job or service for variant coding
STOREDDATA	Read / write stored data
CURRENTDATA	Read current data
MEMORY	Read / write ECU memory
CONTROL	Start/stop routine, request routine results
CALIBRATION	Job or Service for ECU calibration
ROUTINE	Start / stop a routine and / or request routine results
FUNCTION	Execute function like enable/disable normal message transmission or reset ECU
DOWNLOAD	Job or service to request a download data transfer (to the ECU)
UPLOAD	Job or service to request an upload data transfer (from the ECU)
FLASHJOB	Job used for reprogramming of ECU memory
TESTERPRESENT	service to keep an active diagnostic session alive (may only appear once)
DEFINE-DYN-MESSAGE	Service defines a DYN-ID record
READ-DYN-DEFINED-MESSAGE	Service reads data using a DYN-ID
MEMORYREAD	Job or service for reading ECU memory, e.g. Read Memory By Address Service
MEMORYWRITE	Job or service for writing ECU memory, e.g. Write Memory By Address Service
STOREDDATAREAD	Job or service for reading stored data, e.g. Read Data By Identifier Service
STOREDDATAWRITE	Job or service for writing stored data, e.g. Read Write By Identifier Service
VARIANTCODINGREAD	Job or service for reading variant coding data
VARIANTCODINGWRITE	Job or service for writing variant coding data
EVENTREAD	Job or service for interpretation of event triggered ECU responses
EVENTCLEAR	Reset event logic in the ECU to stop event triggered responses
DEFAULT-EVENT-READ	Default interpretation of event triggered ECU responses (may only appear once)
DEFAULT-EVENT-CLEAR	Reset default event logic in the ECU (may only appear once)

Table A.2 — SEMANTIC at PARAM

Pre-defined value	Description
ID	Identifier
SERVICE-ID	Service identifier
SUBFUNCTION	Subfunction of service identifier
CONTROL	Control parameter
DATA	Data
LOCAL-ID	Local identifier
COMMON-ID	Common identifier
CONTROLPARAMETER-ID	Control parameter identifer
DATA-ID	Data identifier
DATAPACKAGE-ID	Data package identifier
PARAMETER-ID	Parameter identifier
DYN-ID	Dynamically defined identifier within a request and a response

Table A.3 — SEMANTIC at TABLE

Pre-defined value	Description
DATA-ID	Data identifier
PACKAGE-ID	Package identifier
PARAMETER-ID	Parameter identifier
LOCAL-ID	Local identifier
COMMON-ID	Common identifier
ADDRESS	Memory address in the ECU

A.2 Values of extendable enumerations

Table A.4 — Enumeration of SYSPARAM at SYSTEM

value	Description
TIMEZONE	Count of minutes to UTC
YEAR	Indication of year
MONTH	Indication of month
DAY	Indication of day
HOUR	Indication of hour (0-23)
MINUTE	Indication of minute (0-59)
SECOND	Indication of second (0-59)
TESTERID	Identification identifier of the tester
USERID	Identification identifier of the user
CENTURY	Indication of century
WEEK	Indication of week

Table A.5 — Enumeration of TYPE at DATABLOCK

value	Description
BOOT	Data block of boot memory
CODE	Data block of code code memory
DATA	Data block of data memory

Table A.6 — Enumeration of TYPE at PROTOCOL

value	Description
ISO_11898_RAW	ISO RAW CAN frames
ISO_14230_3_on_ISO_14230_2	ISO KWP2000 on K Line
ISO_14230_3_on_ISO_15765_2	ISO KWP2000 on CAN
ISO_15765_3_on_ISO_15765_2	ISO UDS on CAN
ISO_15031_5_on_ISO_9141_2	ISO OBD on 9141-2 K/L-Line
ISO_15031_5_on_SAE_J1850VPW	ISO OBD on SAE J1850 VPW
ISO_15031_5_on_ISO_15765_2	ISO OBD on CAN
ISO_15031_5_on_SAE_J1850PWM	ISO OBD on SAE J1850 PWM
ISO_15031_5_on_ISO_14230_2	ISO OBD on KWP2000 K-Line
SAE_J1587_on_SAE_J1708	SAE Truck and Bus on UART
SAE_J1939_73_on_SAE_J1939_21	SAE Truck and Bus on CAN
SAE_J2190_on_ISO_14230_2	SAE J2190 Enhanced Diagnostics on K-Line
SAE_J2190_on_ISO_15765_2	SAE J2190 Enhanced Diagnostics on CAN
SAE_J2190_on_SAE_J1850VPW	SAE J2190 Enhanced Diagnostics on SAE J1850 VPW
SAE_J2190_on_SAE_J1850PWM	SAE J2190 Enhanced Diagnostics on SAE J1850 PWM
SAE_J2610_on_SAE_J2610_SCI	SAE Chrysler SCI on UART

Table A.7 — Enumeration of PARAM-CLASS at COMPARAM

Value	Description
TIMING	Message flow timing parameters, e.g. inter byte time or time between request and response.
INIT	Parameters for initiation of communication e.g. trigger address or wakeup pattern. These parameters must not be overwritten within ECU-Variant layer in any way.
COM	General communication parameter
ERRHDL	Parameter defining the behaviour of the runtime system in case an error occurred, e.g. runtime system could either continue communication after a timeout was detected, or stop and reactivate.
BUSTYPE	This is used to define bustype specific parameters (e.g. baudrate). Most of these parameters affect the physical hardware. These parameters can only be modified by the first LOGICAL-LINK that acquired the physical resource. When a second LOGICAL-LINK is created for the same resource, these parameters that were previously set will be active for the new LOGICAL-LINK.
UNIQUE_ID	This type of ComParam is used to indicate to both the ComLogicalLink and the Application that the information is used for protocol response handling from a physical or functional group of ECUs to uniquely define an ECU response.

Table A.8 — Enumeration "PHYSICAL-LINK-TYPE"

Value	Description
ISO_11898_2_DWCAN	Road vehicles — Controller area network (CAN) — High-speed medium access unit
ISO_11898_3_DWFTCAN	Road vehicles — Controller area network (CAN) — Low-speed, fault tolerant, medium dependent interface
ISO_11992_1_DWCAN	Road vehicles — Interchange of digital information on electrical connections between towing and towed vehicles — Physical layer and data link layer
ISO_9141_2_UART	Road vehicles — Diagnostic systems — CARB requirements for interchange of digital information
ISO_14230_1_UART	Road vehicles — Diagnostic systems — Keyword protocol 2000 — Physical layer
ISO_11898_RAW	Road vehicles — Controller area network (CAN) — Data link layer and physical signalling
SAE_J1850_VPW	Class B Data Communications Network Interface (GM Class 2)
SAE_J1850_PWM	Class B Data Communications Network Interface (Ford SCP)
SAE_J2610_UART	Serial Data Communication Interface (Chrysler SCI bus type)
SAE_J1708_UART	Serial Data Communications Between Microcomputer Systems in Heavy-Duty Vehicle Applications (RS 485 type bus type)
SAE_J1939_11_DWCAN	Recommended Practice for Serial Control and Communications vehicle Network – Physical Layer, 250K bits/s, Twisted Shielded Pair (Dual Wire CAN)
GMW_3089_SWCAN	GMLAN Single Wire CAN Physical and Data Link Layer Specification
XDE_5024_UART	SERIAL DATA BUS FOR COMMUNICATIONS BETWEEN MICROCOMPUTER ASSEMBLIES (UART with Delco Electronics SXR 5V interface)
CCD_UART	SAE 880586 Chrysler Collision Detection Bus Interface (UART with 5V differential interface circuit at 7812.5 bits/s)

[1] SAE J1587:FEB2002 Electronic Data Interchange Between Microcomputer Systems in Heavy-Duty Vehicle Applications

[2] SAE J1708:AUG2004 Serial Data Communications Between Microcomputer Systems in Heavy-Duty Vehicle Applications

[3] SAE J1850:MAY2001 Class B Data Communications Network Interface.

[4] SAE J1939 Recommended Practice for Serial Control and Communications vehicle Network

[5] SAE J2190 Enhanced E/E Diagnostic Test Modes

[6] SAE J2534-1:DEC2004 Recommended Practice for Pass-Thru Vehicle Programming

[7] SAE J2610:APR2002 Serial Data Communication Interface

A.3 Values of non-extendable enumerations

Table A.9 — Enumeration "STANDARDISATION-LEVEL"

value	Description
STANDARD	The communication parameter belonging to a standardized protocol (or defined by the ODX working group) has to be supported by a runtime system implementing this corresponding standardized protocol. This one cannot execute diagnostic data using a protocol not supported by the runtime system.
OEM-SPECIFIC	The communication parameter is part of a non-standardized OEM-specific protocol; nevertheless it is required to be implemented by the runtime system. Diagnostic data using an OEM-specific protocol not supported by the runtime system cannot be executed by the runtime system.
OPTIONAL	This communication parameter does not have to be supported by the runtime system. If a diagnostic service uses an unsupported comparam of this type the comparam can be ignored and the diagnostic service can be executed nevertheless.

Table A.10 — Enumeration "DIAG-CLASS-TYPE"

value	Description
STARTCOMM	Starts the communication.
STOPCOMM	Stops the communication.
VARIANTIDENTIFICATION	Used for identification of the ECU variant.
READ-DYN-DEFINED-MESSAGE	Reads the data via DYN-ID.
DYN-DEF-MESSAGE	Creates a DYN-ID.
CLEAR-DYN-DEF-MESSAGE	Deletes a DYN-ID.

Table A.11 — Enumeration "ADDRESSING"

value	Description
FUNCTIONAL	Only functional addressing is supported.
PHYSICAL	Only physical addressing is supported. Only an individual ECU can be addressed in this mode.
FUNCTIONAL-OR-PHYSICAL	Both modes are supported.

Table A.12 — Enumeration "ROW-FRAGMENT"

value	Description
KEY	
STRUCT	
KEY-AND-STRUCT	

Table A.13 — Enumeration "COMM-RELATION-VALUE-TYPE"

value	Description
CURRENT	The value should be retrieved from the tester/ECU every time when the diag-variable is accessed. The runtime system sends/receives the actual value. This is the default if no VALUE-TYPE is specified.
STORED	When a changed value is retrieved from the tester then the runtime system should store this value for further use. The stored value is sent to the ECU.
STATIC	The value should be retrieved once from the tester/ECU when the diag-variable is accessed for the first time. After that the value can't be changed anymore. An example for a static value is an ECU-internal identifier.
SUBSTITUTED	These values are neither current nor stored but values delivered by the ECU as substituted (or intermediate) values.

Table A.14 — Enumeration "COMPU-CATEGORY"

value	Description
IDENTICAL	Just passes on the input value so that the internal value and the physical one are identical.
LINEAR	CompuMethods of type LINEAR multiply the internal value with a factor and add on an offset. Exactly one COMPU-SCALE has to be defined.
SCALE-LINEAR	Similar to LINEAR. But multiple COMPU-SCALEs can be defined.
TEXTTABLE	The type TEXTTABLE transforms the internal value into a textual expression via mapping defined by this computational method.
COMPUCODE	A computer program coded in Java programming language does the computation.
TAB-INTP	A CompuMethod of type TAB-INTP defines a set of points and the physical value is to be calculated via linear interpolation.
RAT-FUNC	A rational function differs from the linear function in the omission of the restrictions for COMPU-NUMERATORs and COMPU-DENOMINATORs. They can have as many V-values as needed.
SCALE-RAT-FUNC	Similar to RAT-FUNC. But multiple COMPU-SCALEs can be defined.

Table A.15 — Enumeration "DATA-TYPE"

value	Description
A_INT32	32 bit signed integer (range: -21474836248 .. 2147483647)
A_UINT32	32 bit unsigned integer (range: 0 .. 4294967295)
A_FLOAT32	32 bit float (IEEE754 single precision)
A_FLOAT64	64 bit float (IEEE754 double precision)
A_ASCIISTRING	ASCII-coded zero-terminated character field (ISO/IEC 8859-1) with maximum length: 2^32-1 characters (without delimiter 0x00)
A_UTF8STRING	UTF-8 coded zero-terminated character field with maximum length: 2^32-1 characters (without delimiter 0x00)
A_UNICODE2STRING	UNICODE-2-coded zero-terminated character field (ISO/IEC 10646 UCS-2) with maximum length: 2^32-1 characters (without delimiter 0x0000)
A_BYTEFIELD	Byte field with maximum length: 2^32-1

Table A.16 — Enumeration "DATAFORMAT-SELECTION"

value	Description
INTEL-HEX	Intel hex-format
MOTOROLA-S	Motorola hex-format
BINARY	Unformatted data

Table A.17 — Enumeration "ENCODING"

value	Description
BCD-P	Packed BCD format --- A BCD digit is packed in 4 bits (nibble) (47 = 47h).
BCD-UP	Unpacked BCD format --- A BCD digit is mapped to one byte (17 = 0107h).
1C	One's complement
2C	Two's complement
SM	Sign-Magnitude --- The encoding sign-magnitude means the first bit represents the sign (e.g. "1" equals "-"). In the following bits the magnitude of the value is coded.)
UTF-8	
UCS-2	
ISO-8859-1	
ISO-8859-2	
WINDOWS-1252	
NONE	In case the BASE-TYPE-ENCODING is not defined explicitly the "default encoding" has to be applied.

Table A.18 — Enumeration "ENCRYPT-COMPRESS-METHOD-TYPE"

value	Description
A_UINT32	
A_BYTEFIELD	

Table A.19 — Enumeration "IDENT-VALUE-TYPE"

value	Description
A_UINT32	The value of the IDENT should be interpreted as a type A_UINT32.
A_BYTEFIELD	The value of the IDENT should be interpreted as a type A_BYTEFIELD.
A_ASCIISTRING	The value of the IDENT should be interpreted as a type A_ASCIISTRING.

Table A.20 — Enumeration "DIRECTION"

value	Description
DOWNLOAD	Data transfer from tester to ECU
UPLOAD	Data transfer from ECU to tester

Table A.21 — Enumeration "INTERVAL-TYPE"

value	Description
OPEN	The edge value is not to be included.
CLOSED	The edge value is to be included.
INFINITE	Is used if the value defined by a LIMIT is -∞ or +∞

Table A.22 — Enumeration "PHYSICAL-DATA-TYPE"

value	Description
A_INT32	32 bit signed integer (range: -21474836248 .. 2147483647)
A_UINT32	32 bit unsigned integer (range: 0 .. 4294967295)
A_FLOAT32	32 bit float (IEEE754 single precision)
A_FLOAT64	64 bit float (IEEE754 double precision)
A_UNICODE2STRING	UNICODE-2-coded zero-terminated character field (ISO-10646 UCS-2) with maximum length: 2^32-1 characters (without delimiter 0x0000)
A_BYTEFIELD	Byte field with maximum length: 2^32-1

Table A.23 — Enumeration "RADIX"

value	Description
HEX	Hexadecimal number
DEC	Decimal number
BIN	Dual number
OCT	Octal number

Table A.24 — Enumeration "SESSION-SUB-ELEM-TYPE"

value	Description
A_UINT32	The checksum result is of type A_UINT32
A_BYTEFIELD	The checksum result is of type A_BYTEFIELD
A_ASCIISTRING	The checksum result is of type A_ASCIISTRING

Table A.25 — Enumeration "TERMINATION"

value	Description
END-OF-PDU	The termination is done by the end of the PDU.
ZERO	The termination is done by a Byte with the value 00hex.
HEX-FF	The termination is done by a Byte with the value FFhex.

Table A.26 — Enumeration "UNIT-GROUP-CATEGORY"

value	Description
COUNTRY	Group of units, wich are associated to a country-specific unit system.
EQUIV-UNITS	Group of equivalent units, which are used in different countries for the same application domain.

Table A.27 — Enumeration "VALID-TYPE"

value	Description
VALID	Valid area. Current value is within a valid range and can be presented to user.
NOT-VALID	Invalid area. The ECU cannot process the requested data.
NOT-DEFINED	Indicates an area, which is marked in a specification (e.g. as reserved). Shall usually not be set by the ECU but is used by a tester to verify correct ECU behaviour. Also prevents usage of areas during editing process.
NOT-AVAILABLE	Currently invalid area. The value usually is presented by the ECU but can currently not be performed due to e.g. initialisation or temporary problems. This behaviour appears during runtime and cannot be handled while data is edited.

Table A.28 — Enumeration "PIN-TYPE"

value	Description
HI	"High" line/wire of a two wired physical data link
LOW	"Low" line/wire of a two wired physical data link
K	"K" line/wire of a ISO_9141_2_UART or ISO_14230_1_UART physical data link
L	"L" line/wire of a ISO_9141_2_UART or ISO_14230_1_UART physical data link
TX	"High" line/wire of a two wired physical data link
RX	"High" line/wire of a two wired physical data link
PLUS	"Positive (+)" line/wire of a two wired physical data link
MINUS	"Negative (-)" line/wire of a two wired physical data link
SINGLE	"Single" line/wire of a one wired physical data link (e.g. GMW_3089_SWCAN)

Annex B
(normative)

Checker rules

B.1 Overview

The list of checker rules is more than an appendix but important part of the specification. The purpose of the checker rules is to ensure the interchange ability of ODX-data. Note that the compliance to the checker rules is mandatory for ODX compliant data. An application checking the consistency and conformity of ODX-data should create an error message in case of violation of a checker rule. Some of the rule descriptions are classified as "[warning]". In that case the application should report a warning instead.

B.2 Table of checker rules

Table B.1 — Checker rules

ID	Location	Detailed description	Error Level	Runtime relevant
1	LANGUAGE	Every LANGUAGE element must contain a four-letter language code that is specified by xsd:language.	warn	no
3	SHORT-NAME	Every SHORT-NAME value is matched against the regular expression [a-zA-Z0-9_]+.	error	no
4	DIAG-COMM	The PROTOCOLS that are supported by the DIAG-COMM are determined as follows: If at least one PROTOCOL-SNREF exists then the referenced PROTOCOLS are supported. If no PROTOCOL-SNREF exists then all PROTOCOLS within the layer hierarchy of the layer that contains the DIAG-COMM are supported. Every COMPARAM-REF at the DIAG-COMM must refer to at least one of the COMPARAMS in the COMPARAM-SPECS that are referenced by the supported PROTOCOLS.	error	yes
5	PROG-CODE	For every PROG-CODE element the value of the SYNTAX sub-element is matched against the strings "JAVA", "CLASS", and "JAR".	warn	yes
6	PROG-CODE	For every PROG-CODE element the value of the SYNTAX sub-element is matched against the strings "JAVA" and "CLASS". If the value does not match one of these strings then it will be checked if the ENTRYPOINT within the same PROG-CODE exists.	warn	yes
7	PARAM of type MATCHING-REQUEST-PARAM	Check for every DIAG-COMM with a defined request bytelength for every MATCHING-REQUEST-PARAM element if the value of the REQUEST-BYTE-POS added to BYTE-LENGTH is lesser than or equal to the byte length of the request belonging to the same DIAG-COMM.	error	yes

Table B.1 — Checker rules

8	PARAM of type SYSTEM	For every SYSTEM element the value of the SYSPARAM attribute is matched against the strings "TESTERID" and "USERID". If the value does match one of these strings, the BASE-DATA-TYPE at PHYSICAL-TYPE of the refered DOP is checked if it is A_BYTEFIELD. If the value does not match one of these strings, the BASE-DATA-TYPE at PHYSICAL-TYPE of the refered DOP is checked if it is A_UINT32.	error	yes
9	BITLENGTH	Every BIT-LENGTH value is checked if it is greater than 0. If not then an error message is issued that the BIT-LENGTH value must be greater than 0.	error	yes
10	BITPOSITION	Every BIT-POSITION value is checked if it is lesser than 8.	error	yes
11	DATA-OBJECT-PROP	PHYSICAL-TYPE/DISPLAY-RADIX is only allowed if PHYSICAL-TYPE/BASE-DATA-TYPE is defined as a A_UINT32. PHYSICAL-TYPE/PRECISION is only allowed if PHYSICAL-TYPE/BASE-DATA-TYPE is defined as A_FLOAT32 or A_FLOAT64.	error	no
12	DATA-OBJECT-PROP	DIAG-CODED-TYPE/MIN-LENGTH must be lower or equal than DIAG-CODED-TYPE/MAX-LENGTH.	error	yes
14	DATA-OBJECT-PROP	Each range of SCALE-CONSTR must not overlap any other range. Two closed range ends with the same limit are prohibited	error	yes
15	END-OF-PDU-FIELD	END-OF-PDU-FIELD/MIN-NUMBER-OF-ITEMS must be lower or equal than END-OF-PDU-FIELD/MAX-NUMBER-OF-ITEMS.	error	yes
16	MUX	For every CASE element, LOWER-LIMIT/CONTENT must be less or equal to UPPER-LIMIT/CONTENT of the same CASE. 'less or equal' has different meaning for every datatype that can appear for a SWITCH-KEY. Numbers are ordered mathematically.	error	yes
17	DYNAMIC-LENGTH-FIELD	The DATA-OBJECT-PROP referenced by the DETERMINE-NUMBER-OF-ITEMS element: The DATA-OBJECT-PROP's PHYSICAL-TYPE/BASE-DATA-TYPE must be A_UINT32"	error	yes
18	DTCS	A DTC-DOP holds a set of DTCs that may either be referenced via an odxlink or that may be contained within the DTC-DOP. Within this set, every DTC/TROUBLE-CODE must be unique excepting the temporary DTCs.	error	yes
19	ENV-DATA-DESC	For every ENV-DATA-DESC element, the PARAM referenced by PARAM-SNREF must reference a DATA-OBJECT-PROP or a DTC-DOP with PHYSICAL-TYPE/BASE-DATA-TYPE set to A_UINT 32.	error	yes
20	ENV-DATA-DESC	An ENV-DATA-DESC element contains a set of ENV-DATA elements that are either referenced via an odxlink or that may be contained within the ENV-DATA-DESC itself. For every ENV-DATA-DESC element this set contains no more than one ENV-DATA, which contains an ALL-VALUE element.	error	yes
21	ENV-DATA-DESC	An ENV-DATA-DESC element contains a set of ENV-DATA elements that are either referenced via an odxlink or that may be contained within the ENV-DATA-DESC itself. DTC-VALUE must be unique among ENV-DATAs of a common ENV-DATA-DESC.	error	yes

Table B.1 — Checker rules

22	DIAG-VARIABLE	Case 1: COMM-RELATION refers to a DIAG-SERVICE: The parameter referenced by IN-PARAM-IF-SNREF must exist in the DIAG-SERVICE's request and the parameter referenced by OUT-PARAM-IF-SNREF must exist in at least one of the DIAG-SERVICE's POS-RESPONSEs. Case 2: COMM-RELATION referst to a SINGLE-ECU-JOB: The parameter referenced by IN-PARAM-IF-SNREF must exist amongst the SINGLE-ECU-JOB's INPUT-PARAMs and the parameter referenced by OUT-PARAM-IF-SNREF must exist amongst the SINGLE-ECU-JOB's OUTPUT-PARAMs.	error	yes
23	MATCHING-PARAMETER	MATCHING-PARAMETER references a DIAG-COMM and a OUT-PARAM-IF via SNREF. The SHORT-NAME of the OUT-PARAM-IF-SNREF must exist in all positive responses in the case of DIAG-SERVICE.	warn	yes
24	COMPARAM-SPEC	Each ODXLINK must reference an element within the same COMPARAM-SPEC structure. This means, the ODXLINK must not have a DOCREF element.	error	yes
25	MATCHING-COMPONENT	MATCHING-COMPONENT references a DIAG-COMM or MULTIPLE-ECU-JOB via odx-link and a OUT-PARAM-IF via SNREF. The SHORT-NAME of the OUT-PARAM-IF-SNREF must exist in all positive responses in the case of DIAG-SERVICE or in one OUTPUT-PARAM in CASE of the SINGLE-ECU-JOB or MULTIPLE-ECU-JOB.	warn	yes
27	LOGICAL-LINK	In case a BASE-VARIANT or a FUNCTIONAL-GROUP inherits from multiple PROTOCOLs, a LOGICAL-LINK for this BASE-VARIANT or FUNCTIONAL-GROUP must also refer to exactly one of the PROTOCOL layers.	error	yes
28	IDENT-DESC	IDENT-DESC references a DIAG-COMM and a OUT-PARAM-IF via shortname reference. In case of a DIAG-SERVICE is referenced: The referenced parameter must exist in all positive responses of the DIAG-SERVICE. In case of a SINGLE-ECU-JOB is referenced: The OUT-PARAM must exist at the SINGLE-ECU-JOB.	warn	yes
29	LAYER-REF within ECU-MEM-CONNECTOR	Only ECU-VARIANT and BASE-VARIANT are valid types for reference by the LAYER-REF attribute of ECU-MEM-CONNECTOR. Check that the list of LAYERs refered to only contain these types.	error	yes
30	MULTIPLE-ECU-JOB-SPEC	Iterate through the list of DOP-BASE objects in the associated DIAG-DATA-DICTIONARY-SPEC. Check that in all cases the IS-HIGHLOW-BYTE-ORDER attribute is not set.	warn	yes

Table B.1 — Checker rules

31	general	Targets of odx-links must be contained in the data pool of the current layer. The data pool contains all the instances of the parent layer hierarchy and the imported layers and in case of an ECU-VARIANT all other ECU-VARIANTS belonging to the same BASE-VARIANT.	error	yes
32	general	Direct and indirect cycles of odx-links are not allowed. Direct cyclic references occur when object A refers B and B refers A. Indirect cyclic references occur when A occurs in the transitive closure over A's reference chain. In both cases this would result in an infinite reference loop.	error	yes
33	general	All references may only reference the target as specified by the ODX UML model.	error	yes
35	PARAM of type VALUE	PARAM of type VALUE has an optional PHYSICAL-DEFAULT-VALUE element and a DOP-REF/DOP-SNREF which can reference both a COMPLEX-DOP or a DATA-OBJECT-PROP. The optional PHYSICAL-DEFAULT-VALUE can only exist if the referenced DOP is a DATA-OBJECT-PROP.	error	no
36	SHORT-NAME	Every SHORT-NAME must not have more than 128 characters.	error	yes
38	DTC-DOP	The BASE-DATA-TYPE at the PHYSICAL-TYPE must be set to A_UINT32.	error	yes
39	general	At least one potocol layer must exist in the inheritance hierarchy of DIAG-LAYERS.	error	yes
41	DIAG-SERVICE	A PARAM with SEMANTIC=SERVICE-ID shall occur at most once within a REQUEST or POS-RESPONSE, respectively.	error	no
42	DIAG-CODED-TYPE	If BASE-DATA-TYPE is A_BYTEFIELD, A_ASCIISTRING, A_UTF8STRING the IS-HIGHLOW-BYTE-ORDER must not be present except DIAG-CODED-TYPE is of type LEADING-LENGTH-INFO-TYPE.	error	no
43	DATA-OBJECT-PROP	If INTERNAL-CONSTR is present within element DATA-OBJECT-PROP, then DIAG-CODED-TYPE/BASE-DATA-TYPE shall be a ordered data type (A_BYTEFIELD, A_INT32, A_UINT32, A_FLOAT32 or A_FLOAT64).	error	yes
44	MUX/SWITCH-KEY	DATA-OBJECT-PROP/DIAG-CODED-TYPE in element SWITCH-KEY shall be of type STANDARD-LENGTH-TYPE	error	yes
46	MUX/CASE	If a MUX element has no CASES sub-element then it must have a DEFAULT-CASE sub-element.	error	yes
47	COMPARAM-REF	Within every COMPARAM-REFS there shall not be more than one COMPARAM-REF referencing the same COMPARAM and the same PROTOCOL.	error	yes
48	NOT-INHERITED-DIAG-COMM	For each entry in a NOT-INHERITED-DIAG-COMMS list of a PARENT-REF the matching DIAG-COMM is retrieved from the parent layer referenced by PARENT-REF. If the parent layer inherits the DIAG-COMM from another layer then it is retrieved from the layer that the DIAG-COMM is defined in. These DIAG-COMM's attribute IS-MANDATORY must be false.	error	no

Table B.1 — Checker rules

49	DIAG-COMM	For each DIAG-COMM in a layer the parent hierarchy is searched for a DIAG-COMM that has the same SHORT-NAME as the overwriting DIAG-COMM. If a DIAG-COMM with the same SHORT-NAME is found then this DIAG-COMM's attribute IS-FINAL must be false.	error	no
50	DIAG-LAYER-CONTAINER	Only the following PARENT-REF relationships are allowed: PROTOCOL -PARENT-REF-> ECU-SHARED-DATA FUNCTIONAL-GROUP -PARENT-REF-> ECU-SHARED-DATA FUNCTIONAL-GROUP -PARENT-REF-> PROTOCOL BASE-VARIANT -PARENT-REF-> ECU-SHARED-DATA BASE-VARIANT -PARENT-REF-> PROTOCOL BASE-VARIANT -PARENT-REF-> FUNCTIONAL-GROUP ECU-VARIANT -PARENT-REF-> ECU-SHARED-DATA ECU-VARIANT -PARENT-REF-> BASE-VARIANT In all cases, the PARENT-REF source layer can have multiple PARENT-REFs to DIAG-LAYER types given above. Exception: An ECU-VARIANT can only have one BASE-VARIANT as a parent.	error	yes
51	REQUEST	Every bit within a request must be covered by exactly one PARAM.	error	no
52	COMPU-METHOD of type TAB-INTP	Each COMP-INTERNAL-TO-PHYS/COMPU-SCALE of type TAB-INTP must contain exactly one LOWER-LIMIT with CONTENT of numerical type (no empty string) and no UPPER-LIMIT.	error	yes
53	COMPU-METHOD of type COMPUCODE	The sub-element COMPU-INTERNAL-TO-PHYS must at most have the subelement PROG-CODE.	error	yes
54	UNIT	PHYSICAL-DIMENSION-REF, FACTOR-SI-TO-UNIT, OFFSET-SI-TO-UNIT: If one of these elements exists within the UNIT, the both remaining elements must also exist within the UNIT.	error	yes
55	DIAG-LAYER	An ECU-SHARED-DATA must not both, be imported and inherited by DIAG-LAYERs within the inheritance chain. If a DIAG-LAYER A inherits an ECU-SHARED-DATA E, E cannot be imported by A or any DIAG-LAYER in the inheritance tree (that is all transitive parents and children of A with respect to value inheritance) of A If a DIAG-LAYER A imports an ECU-SHARED-DATA E, E cannot be inherited by A or any DIAG-LAYER in the inheritance tree of A	error	yes
56	DIAG-LAYER	Within the intersection of a specialisation level and a namespace (see chapter 6.3.2.4.1) and a boundary (see section 6.3.15.3, Table 14), any SHORT-NAME shall uniquely define a data object. This rule extends to objects referenced via odx-link within proxy wrappers as well. The ODX data model contains the following proxy wrappers: DIAG-COMM-PROXY, DTC-PROXY, ENV-DATA-PROXY and DIAG-VARIABLE-PROXY, TABLE-ROW.	error	yes
57	DATA-OBJECT-PROP	Each range of SCALE-CONSTR must fit into the INTERNAL-CONSTR range.	error	yes

Table B.1 — Checker rules

58	ACCESS-LEVEL COMM-RELATION DIAG-SERVICE IDENT-DESC MATCHING-PARAMETER SESSION-DESC	The enclosed SHORT-NAME or ODX-LINK reference element may only refer to DIAG-COMMs that are either defined or referenced by ODX-LINK within at least one of the DIAG-COMMS wrappers of the enclosing DIAG-LAYERs value-inheritance scope or of the same DIAG-LAYER. The names of the reference elements are: In case of ACCESS-LEVEL: DIAG-COMM-SNREF; In case of COMM-RELATION: DIAG-COMM-REF or DIAG-COMM-SNREF; In case of DIAG-SERVICE: RELATED-DIAG-COMM-REF; In case of IDENT-DESC: DIAG-COMM-SNREF; In case of MATCHING-PARAMETER: DIAG-COMM-SNREF; In case of SESSION-DESC: DIAG-COMM-SNREF;	error	yes
59	COMPU-METHOD of type IDENTICAL	Within a DATA-OBJECT-PROP containing a COMPU-METHOD of COMPU-CATAGORY IDENTICAL the PHYSICAL-TYPE / BASE-DATA-TYPE and the DIAG-CODED-TYPE / BASE-DATA-TYPE must be identical. Exception if the DIAG-CODED-TYPE / BASE-DATA-TYPE is A_ASCIISTRING or A_UTF8STRING the PHYSICAL-TYPE / BASE-DATA-TYPE is A_UNICODE2STRING.	error	yes
60	COMPU-METHOD of type LINEAR, TAB-INTP, SCALE-LINEAR, RAT-FUNC or SCALE-RAT-FUNC	Only datatypes A_INT32, A_UINT32, A_FLOAT32 or A_FLOAT64 are alowed as internal or physical type.	error	yes
61	COMPU-METHOD of type TEXTTABLE	Within a DATA-OBJECT-PROP containing a COMPU-METHOD of COMPU-CATAGORY TEXTTABLE the PHYSICAL-TYPE / BASE-DATA-TYPE must be A_UNICODE2STRING.	error	yes
62	COMPU-METHOD of type TEXTTABLE	Within a DATA-OBJECT-PROP containing a COMPU-METHOD of COMPU-CATAGORY TEXTTABLE the DIAG-CODED-TYPE / BASE-DATA-TYPE must be A_INT32, A_UINT32, A_ASCIISTRING, A_UTF8STRING, A_UNICODE2STRING or A_BYTEFIELD.	error	yes
63	DOP with COMPU-METHOD of type IDENTICAL	Within a DATA-OBJECT-PROP containing a COMPU-METHOD of COMPU-CATAGORY IDENTICAL no COMPU-INTERNAL-TO-PHYS and no COMPU-PHYS-TO-INTERNAL objects are allowed	warn	no
64	COMPU-METHOD of type LINEAR or RAT-FUNC	Within COMPU-METHOD of COMPU-CATAGORY LINEAR or RAT-FUNC only one COMPU-SCALE object is allowed	error	yes
68	IDENT-DESC	The referenced parameter (OUT-PARAM-IF) must refer to a DATA-OBJECT-PROP	error	no
69	IDENT-IF	The values of attributes TYPE at all IDENT-VALUEs of one EXPECTED-IDENT must be identical	error	no

Table B.1 — Checker rules

70	IDENT-DESC	The values of the attributes TYPE at IDENT-VALUEs of the referenced IDENT must match the PHYSICAL-TYPE/BASE-DATA-TYPE defined in the DOP of the referenced parameter (IN-PARAM-IF or OUT-PARAM-IF)	error	no
71	PHYS-SEGMENT of type ADDRDEF-PHYS-SEGMENT	The START-ADDRESS must contain a value less or equal to END-ADDRESS.	error	no
72	CHECKSUM	If the CHECKSUM contains SOURCE-END-ADDRESS, the SOURCE-START-ADDRESS must contain a value less or equal the SOURCE-END-ADDRESS.	error	no
73	FILTER of type ADDRDEF-FILTER	FILTER-START must contain a value less or equal to FILTER-END	error	no
74	SEGMENT	If the SEGMENT contains SOURCE-END-ADDRESS, then SOURCE-START-ADDRESS must contain a value less or equal to SOURCE-END-ADDRESS.	error	no
75	SECURITY-METHOD, FW-CHECKSUM, VALIDITY-FOR, FW-SIGNATURE, CHECKSUM-RESULT, IDENT-VALUE, ENCRYPT-COMPRESS-METHOD	The CONTENT of these elements must correspond with the datatype defined by the attribute TYPE. For example: If the TYPE is A_UINT32, then the CONTENT must be interpretable as an A_UINT32 value.	error	no
76	DATABLOCK	Address ranges of two different SEGMENTs in one DATABLOCK must not overlap.	error	no
77	DATABLOCK	Address ranges of two different FILTERs in one DATABLOCK must not overlap.	error	no
78	DATABLOCK	In case of hex formatted data (INTEL-HEX, MOTOROLA-S) and if at least one SEGMENT and at least one FILTER are defined, each SEGMENT must correspond to existing data after applying the FILTERs. This means that each SEGMENT must lie in the interval of one or more adjacent FILTERs defined in the same DATABLOCK	error	no
79	FILTER of type SIZEDEF-FILTER	FILTER-SIZE must contain a value greater than zero.	error	no
80	CHECKSUM	If the CHECKSUM contains UNCOMPRESSED-SIZE, then UNCOMPRESSED-SIZE must contain a value greater than zero.	error	no

Table B.1 — Checker rules

81	SEGMENT	If the SEGMENT contains UNCOMPRESSED-SIZE, then UNCOMPRESSED-SIZE must contain a value greater than zero.	error	no
82	PHYS-SEGMENT of type SIZEDEF-PHYS-SEGMENT	SIZE must contain a value greater than zero.	error	no
83	PHYS-MEM	Address ranges of two different PHYS-SEGMENTs in one PHYS-MEM must not overlap	error	no
84	DATAFILE	The wildcards '*' and '?' can only be used in DATAFILE/CONTENT if attribute LATEBOUND-DATAFILE=true	error	no
85	TABLE	If KEY-DOP is defined it must refer to a simple DATA-OBJECT-PROP.	error	yes
86	TABLE	KEY within TABLE-ROW must match with the physical data type of the DATA-OBJECT-PROP referenced by KEY-DOP-REF.	error	yes
87	TABLE	TABLE-KEY in a response structure must reference a TABLE	error	yes
89	TABLE	The TABLE-STRUCT parameter must reference a parameter of type TABLE-KEY in the same request or response. TABLE-STRUCT and TABLE-KEY must reside in the same STRUCTURE if one of them is an element of a STRUCTURE..	error	yes
90	TABLE	Parameter TABLE-ENTRY can only be used in a STRUCTURE referenced by a TABLE-ROW.	error	yes
91	DIAG-COMM	A DIAG-COMM is able to refer to FUNCT-CLASSes defined in different Diag-Layers. The SHORT-NAMEs of the FUNCT-CLASSes referenced by one DIAG-COMM must be unique.	error	no
92	SESSION-DESC	The SHORT-NAMEs of all FLASH-CLASSes referenced by one SESSION-DESC must be unique.	error	no

Table B.1 — Checker rules

93	REQUEST	COMPU-METHODS used in REQUEST must be invertible / unambigous. 1. IDENTICAL Condition for invertibility: Always 2. LINEAR Condition for invertibility: VN1 != 0 or COMPU-INVERSE-VALUE must be defined. 3. SCALE-LINEAR Condition for invertibility: The transition of neighboring intervalls is strictly monotonic, in particular in case of diskret values. Adjacent COMPU-SCALES must have the same values on their common boundaries AND the VN1 of all intervals must have the same sign or must be 0. If VN1 = 0 then COMPU-INVERSE-VALUE must be specified. 4. RAT-FUNC Inversion is not allowed. 5. SCALE-RAT-FUNC Inversion is not allowed. 6. TEXT-TABLE Condition for invertibility: If the TextTable is not deterministically invertible COMPU-INVERSE-VALUE must be defined for every COMPU-SCALE. 7. TAB-INTP Condition for invertibility: Always: The first COMPU-SCALE matching the COMPU-CONST is used to calculate the reverse value. 8. COMPU-CODE Condition for invertibility: COMPU-PHYS-TO-INTERNAL must be specified	error	yes
96	PROTOCOL	PROTOCOL-SNREFS must not exist at DIAG-COMMs within PROTOCOL-Layer	error	yes
97	MUX	The cases belonging to the same MUX have one defined type of SWITCH-KEY. The datatype of the SWITCH-KEY is determined by the PHYSICAL-DATA-TYPE of the DATA-OBJECT-PROP assigned to the SWITCH-KEY. The CONTENT of all CASEs LIMIT elements must correspond with this datatype of the SWITCH-KEY. For example: If the PHYSICAL-DATA-TYPE/BASE-DATA-TYPE of the SWITCH-KEY is A_INT32, then all CASE/LIMIT/CONTENT elements belonging to the same MUX as this SWITCH-KEY must be interpretable as an A_INT32 value.	error	yes
98	MUX	If the CONTENT of LOWER- and UPPER-LIMIT contains a value of data type A_UNICODE2STRING in context of MUX/CASE both values must be identical.	error	yes
99	TABLE	The value of TABLE-ROW/KEY must be unique inside a TABLE. (including referenced TABLE-ROWs)	error	yes
100	MATCHING-PARAMETER	The DIAGNOSTIC-CLASS of a DIAG-COMM referenced by MATCHING-PARAMETER must be "VARIANTIDENTIFICATION".	warn	no
103	PARENT-REF	The target layer of a PARENT-REF must comply with the PARENT-REF's xsi:type-attribute: xsi:type = PROTOCOL-REF implies that target layer must be a PROTOCOL layer xsi:type = FUNCTIONAL-GROUP-REF implies that target layer must be a FUNCTIONAL-GROUP layer xsi:type = BASE-VARIANT-REF implies that target layer must be a BASE-VARIANT layer xsi:type = ECU-SHARED-DATA-REF implies that target layer must be an ECU-SHARED-DATA layer	error	yes

Table B.1 — Checker rules

104	DIAG-LAYER	Within one DIAG-LAYER DIAG-COMM with the values STARTCOMM or STOPCOMM as DIAGNOSTIC-CLASS may only exist once, regardless of whether they are inherited, imported or locally defined.	error	yes
105	DIAG-LAYER	Within one DIAG-LAYER only one DIAG-COMM with SEMANTIC = DEFAULT-FAULT-READ may exist, regardless of whether it is inhertied, imported or locally defined.	warn	yes
107	SESSION	It is prohibited to reference one DATABLOCK more than once within the same SESSION.	error	yes
108	DIAG-COMM	MATCHING-REQUEST-PARAM and DYNAMIC may only be used in a response.	error	yes
110	MATCHING-PARAMETER	Case 1: DIAG-COMM-SNREF refers to a DIAG-SERVICE: The parameter referenced by OUT-PARAM-IF-SNREF must exist in at least one of the DIAG-SERVICE's POS-RESPONSEs. Case 2: DIAG-COMM-SNREF refers to a SINGLE-ECU-JOB: The parameter referenced by OUT-PARAM-IF-SNREF must exist amongst the SINGLE-ECU-JOB's OUTPUT-PARAMs. (Remark: This rule is to be evaluated on the BASE-VARIANT.)	error	yes
112	DATA-OBJECT-PROP	In case DATA-OBJECT-PROP has a UNIT-REF to a UNIT and UNIT references a FACTOR-SI-TO-UNIT or OFFSET-SI-TO-UNIT, then DATA-OBJECT-PROP/PHYSICAL-TYPE/BASE-DATA-TYPE shall be numeric (A_INT32, A_UINT32, A_FLOAT32 or A_FLOAT64).	error	yes
113	PROG-CODE	Within one SINGLE-ECU-JOB and one MULTIPLE-ECU-JOB the contained PROG-CODE elements must be disjoint regarding their value for the SYNTAX attribute. JAVA, CLASS and JAR exclude each other.	warn	yes
114	DIAG-CODED-TYPE	For DIAG-CODED-TYPE with BASE-DATA-TYPE in {A_INT32, A_UINT32, A_FLOAT32, A_FLOAT64} only STANDARD-LENGTH-TYPE and PARAM-LENGTH-INFO-TYPE can be used.	error	yes
115	LENGTH-KEY-REF	LENGTH-KEY-REF is only allowed to reference PARAM of type LENGTH-KEY within the same PDU	error	yes
116	LENGTH-KEY	Within PARAM of type LENGTH-KEY only simple DOPs with PHYSICAL-TYPE/BASE-DATA-TYPE = A_UNIT32 are allowed.	error	yes
117	COMPU-METHOD of type TEXTTABLE	If the DIAG-CODED-TYPE of a DATA-OBJECT-PROP is defined as a string type the definition of a range for COMPU-METHOD/CATEGORY[TEXTTABLE]/COMPU-SCALES/COMPU-SCALE is not allowed. In this case LOWER-LIMIT/CONTENT must be identical with UPPER-LIMIT/CONTENT.	error	yes
118	PARAM of type RESERVED	PARAM of type RESERVED may only specify CODED-VALUE if DIAG-CODED-TYPE is specified and vice versa (CODED-VALUE and DIAG-CODED-TYPE must only be specified together)	error	yes
119	DYN-DEFINED-SPEC	Within one DYN-DEFINED-SPEC all referenced TABLEs with the same SEMANTIC shall not contain multiple TABLE-ROWs with the same KEY.	error	yes

Table B.1 — Checker rules

120	SHORT-NAME	A SHORT-NAME shall not begin with the prefix "GEN3D_"	error	yes
122	INTERNAL-CONSTR SCALE-CONSTR COMPU-SCALE MUX/CASES/CASE	LOWER-LIMIT/CONTENT must be less or equal to UPPER-LIMIT/CONTENT. If the CONTENT values are equal the INTERVAL-TYPE of both must be set to CLOSED.	error	yes
123	PARAM of type VALUE	The value of PHYSICAL-DEFAULT-VALUE must be interpretable with the type of the referenced DATA-OBJECT-PROP/PHYSICAL-DATA-TYPE/BASE-DATA-TYPE.	error	yes
124	PARAM of type VALUE	The value of PHYSICAL-DEFAULT-VALUE must be within the range of the referenced DATA-OBJECT-PROP/INTERNAL-CONSTR	error	yes
125	PARAM of type PHYS-CONST	The value of PHYS-CONSTANT-VALUE must be interpretable with the type of the referenced DATA-OBJECT-PROP/PHYSICAL-DATA-TYPE/BASE-DATA-TYPE.	error	yes
126	PARAM of type PHYS-CONST	The value of PHYS-CONSTANT-VALUE must be within the range of the referenced DATA-OBJECT-PROP/INTERNAL-CONSTR	error	yes
127	LONG-NAME	Every LONG-NAME must not have more than 255 characters.	error	yes
128	Param of type SYSTEM	Value of SYSPARAM must match one of the values in the predefined value set.	warn	yes
129	MUX	The STRUCTUREs referenced by the CASEs must have IS-VISIBLE=false	error	yes
130	COMPU-METHOD	LOWER-LIMIT must be defined in any case of a COMPU-SCALE definition except the COMPU-METHOD is of CATEGORY LINEAR or RAT-FUNC, where both LIMITs can be omitted.	error	yes
131	DDLID-DEF-MODE-INFO	A DDLID-DEF-MODE-INFO must reference services of DIAGNOSTIC-CLASS CLEAR-DYN-DEF-MESSAGE via odx-link CLEAR-DYN-DEF-MESSAGE-REF, and services of DIAGNOSTIC-CLASS READ-DYN-DEF-MESSAGE via odx-link READ-DYN-DEF-MESSAGE-REF, and services of DIAGNOSTIC-CLASS DYN-DEF-MESSAGE via odx-link DYN-DEF-MESSAGE-REF.	error	yes
133	DIAG-VARIABLE	A DIAG-VARIABLE/COMM-RELATION/IN-PARAM-IF-SNREF and also a DIAG-VARIABLE/COMM-RELATION/OUT-PARAM-IF-SNREF shall not reference a PARAM that is indirectly contained within a FIELD.	error	yes
134	MATCHING-PARAMETER MATCHING-COMPONENT IDENT-DESC	OUT-PARAM-IF-SNREF shall neither reference a PARAM referencing a COMPLEX-DOP nor a PARAM that is direclty or indirectly contained within a FIELD.	error	yes

Table B.1 — Checker rules

135	DIAG-LAYER	If an element of SHORT-NAME A is declared somewhere in the transitive closure of parent relationships of a DIAG-LAYER D, but A is not valid in D through inheritance (because it is eliminated somewhere within the inheritance relationships), A cannot be reused in D. If A is valid in D through inheritance, it can be reused to overwrite the inherited element (except that inherited element has IS-FINAL set to true). If A is not declared anywhere in the transitive closure of parent relationships of a DIAG-LAYER D, A can be freely used in D.	error	yes
136	NOT-INHERITED-DIAG-COMM	If an IS-MANDATORY DIAG-COMM of SHORT-NAME A is inherited into a DIAG-LAYER D, it may not be completely eliminated. In a single-inheritance case it means it cannot be part of any NOT-INHERITED list amongst the transitive closure of inheritance relationships of D; in a multiple-inheritance case, where element with SHORT-NAME A is inherited from multiple sources, it means that exactly one inheritance path must remain open. If some of the inherited elements of SHORT-NAME A are not mandatory, the open inheritance path must inherit one of the mandatory elements into D.	error	yes
138	COMPU-METHOD of type TEXTTABLE	COMPU-INVERSE-VALUE of all the COMPU-SCALEs with same physical string value must be identical.	warn	yes
139	LOGICAL-LINK	A LOGICAL-LINK can either reference a BASE-VARIANT alone or a PROTOCOL alone or a PROTOCOL and a BASE-VARIANT or a PROTOCOL and a FUNCTIONAL-GROUP. All other combinations are invalid.	error	yes
140	COMPU-SCALES	The COMPU-SCALEs LIMITs have to cover at least the complete valid INTERNAL-CONSTR/SCALE-CONSTR ranges. In case of a numerical data type the valid range is defined by the intersection of all the valid SCALE-CONSTR and the value range of the data type.	error	yes
141	PARAM	PARAMs may only reference the kinds of DOPs as specified in the Table named "Positioning of PARAM and DOP types" depending on their respective class.	error	yes
142	general	A SERVICE with ADDRESSING=FUNCTIONAL can only be defined within a PROTOCOL, a FUNCTIONAL-GROUP or an ECU-SHARED-DATA layer.	error	no
143	SESSION-DESC	SESSION-SNREF must reference to a SESSION that exists within the ECU-MEM referenced by the ECU-MEM-CONNECTOR the SESSION-DESC is contained in.	error	yes
145	general	In case if DIAG-CODED-TYPE is A_FLOAT32 the BIT-LENGTH must be equal to 32. In case of A_FLOAT64, the BIT-LENGTH must be equal to 64. That means that the DIAG-CODED-TYPE must be of type STANDARD-LENGTH-TYPE if BASE-DATA-TYPE is set to A_FLOATxx.	error	yes
146	PROG-CODE	The wildcards '*' and '?' can only be used in CODE-FILE if attribute LATEBOUND-CODE-FILE equals "true".	error	yes
147	PROG-CODE	LATEBOUND-CODE-FILE set to “true” is only allowed if SYNTAX is equal to CLASS or JAR.	error	yes
149	IDENT-DESC	IDENT-IF-SNREF must reference to a IDENT-IF that exists within the namespace of the ECU-MEM referenced by the ECU-MEM-CONNECTOR the IDENT-DESC is contained in.	error	yes

Table B.1 — Checker rules

151	COMPU-METHOD	Each range of COMPU-SCALE must not overlap any other range.	error	yes
152	DIAG-SERVICE	If the POS-RESPONSE-SUPPRESSABLE element is defined at a DIAG-SERVICE the parameter referenced by CODED-CONST-SNREF must exist within the request of the DIAG-SERVICE containing the POS-RESPONSE-SUPPRESSABLE element.	error	yes
153	COMPARAM	If CPUSAGE = "APPLICATION", STANDARDIZATION-LEVEL must be OEM-SPECIFIC	error	yes
154	PARAM of type PHYS-CONST or of type VALUE, INPUT-PARAM	In case of DOP/COMPUMETHOD/TEXTTABLE the values of PHYSICAL-DEFAULT-VALUE or PHYS-CONSTANT-VALUE must match case-sensitively with an entry of the corresponding texttable.	error	yes
156	STRUCTURE	The optional attribute BYTE-SIZE gives the size of the whole structure in bytes. It must not be less than the total size of the included parameters. If the STRUCTURE has any dynamic components (e.g. MUX or any FIELD with dynamic length) the BYTE-SIZE must not be specified.	warning	yes
157	ODX-LINK	DOCREF and DOCTYPE must be specified either jointly or not at all	error	yes
158	ScopeOutOfXml ODX-CATAGORY	The content of ODX files must be in accordance to their file extension.	error	yes
159	ScopeOutOfXml PROG-CODE	If the attribute LATEBOUND-CODE-FILE equals "false" the referenced file must exist.	error	yes
160	ScopeOutOfXml DATAFILE	If the attribute LATEBOUND-DATA-FILE equals "false" the referenced file must exist.	error	yes
161	INPUT-PARAM	The optional PHYSICAL-DEFAULT-VALUE can only exist if the referenced DOP is a DATA-OBJECT-PROP.	error	yes
162	INPUT-PARAM	The value of PHYSICAL-DEFAULT-VALUE must be interpretable with the type of the referenced DATA-OBJECT-PROP/PHYSICAL-DATA-TYPE/BASE-DATA-TYPE.	error	yes
163	INPUT-PARAM	The value of PHYSICAL-DEFAULT-VALUE must be within the range of the referenced DATA-OBJECT-PROP/INTERNAL-CONSTR including the INTERNAL-CONSTR/SCALE-CONSTRs validity ranges.	error	yes
165	COMPU-METHOD	If the CATEGORY is LINEAR or SCALE-LINEAR the number of V at COMPU-SCALE/COMPU-NUMERATOR must be 1 or 2 and the number of V at COMPU-DENOMINATOR must be 0 or 1. If the CATEGORY is RAT-FUNC, SCALE-RAT-FUNC the number of V at COMPU-SCALE/COMPU-NUMERATOR must be greater than 0. If DENOMINATOR value is missing, it is set to 1 per default.	error	yes
166	general	If a diagnostic object defined within a ECU-SHARED-DATA is imported into another DIAG-LAYER all the shortname references contained within the imported data must have a target valid within the importing DIAG-LAYER.	error	yes
169	DIAG-COMM	The SHORT-NAMEs of the FUNCT-CLASSes referenced by one DIAG-COMM must be unique.	error	yes
171	DIAG-LAYER	Every DIAG-LAYER may only contain one STATE-CHART of every CATEGORY.	error	yes
173	COMPU-METHOD	If a COMPU-METHOD used within a REQUEST contains a COMPU-DEFAULT-VALUE, this COMPU-DEFAULT-VALUE must contain a COMPU-INVERSE-VALUE	error	

Table B.1 — Checker rules

174	Serviceparameter	READ-SERVICE-PARAM/IN-PARAM-IF-SNREF and WRITE-SERVICE-PARAM/IN-PARAM-IF-SNREF may not point to a parameter that is of type PHYS-CONST or CODED-CONST.	error	
175	ServiceDataConnector	All references to PARAM-IFs on this sheet need to point to parameters that reside within a request or response of the DIAG-COMM referenced by READ-SERVICE-SNREF or WRITE-SERVICE-SNREF. !! Rule needs to be formulated more precisely and be aligned with a couple of similar checker rules that already exist.	error	
176	ServiceDataConnector	The SERVICE-DATA-CONNECTOR/READ-SERVICE-SNREF and SERVICE-DATA-CONNECTOR/WRITE-SERVICE-SNREF shall only reference DIAG-COMMs that reside within Variants referenced by CONFIG-DATA/BASE-VARIANT-DATA for the CONFIG-DATA element in which the SERVICE-DATA-CONNECTOR is contained through a CONFIG-RECORD.	error	
177	Serviceparameter	READ-SERVICE-PARAM/IN-PARAM-IF-SNREF and WRITE-SERVICE-PARAM/IN-PARAM-IF-SNREF need to reference a parameter with a DATA-OBJECT-PROP (formerly known as SIMPLE-DOP).	error	
178	Parameter	All parameters of statically undetermined length (e.g. parameters with MIN-MAX-LENGTH type, LEADING-LENGTH-INFO or of PARAM-LENGTH-INFO-TYPE) need to be byte-aligned (fully cover a set of 1 to many bytes). For all parameters with COMPLEX-DOPs associated the same constraint applies.	error	
179	TABLE	If a TABLE A references a TABLE-ROW from TABLE B, the KEY-DOP-REF of A shall point to the identical DATA-OBJECT-PROP element as the KEY-DOP-REF of B or B has no KEY-DOP-REF at all.	error	
180	FUNCTION-DICTIONARY-SPEC	FUNCTION-DICTIONARY-SPEC: The BASE-VARIANTs and ECU-VARIANTs referenced by a VEHICLE-INFORMATION-CONNECTOR must be valid within the VEHICLE-INFORMATION referenced from this VEHICLE-INFORMATION-CONNECTOR, if such a VEHICLE-INFORMATION is referenced.	error	
181	FUNCTION-DICTIONARY-SPEC	BASE-FUNCTION-NODE/FUNCTION-OUT-PARAM/OUT-PARAM-IF must exist within the POS-RESPONSE PARAMS or OUT-PARAMs of a MULTIPLE-ECU-JOB, DIAG-COMM or DIAG-VARIABLE referenced through BASE-FUNCTION-NODE/VEHICLE-INFORMATION-CONNECTOR/DIAG-LAYER-CONNECTOR/EXECUTABLE-REF. The same applies for IN-PARAM-REF.	error	
182	FUNCTION-DICTIONARY-SPEC	If BASE-FUNCTION-NODE/FUNCTION-OUT-PARAM/OUT-PARAM-IF or BASE-FUNCTION-NODE/FUNCTION-IN-PARAM/IN-PARAM-IF exists, a MULTIPLE-ECU-JOB, DIAG-COMM or DIAG-VARIABLE must also exist within BASE-FUNCTION-NODE/VEHICLE-INFORMATION-CONNECTOR/DIAG-LAYER-CONNECTOR/EXECUTABLE-REF.	error	
183	BYTE-POSITION	If DIAG-CODED-TYPE/BASE-DATA-TYPE is "ASCIISTRING", the length of the PARAM referring to this DIAG-CODED-TYPE must have a BIT-LENGTH that is a multiple of 8.	error	
184	BIT-MASK	The BIT-MASK length must match the length of the parameter. The BIT-MASK is then applied to the whole value.	error	

Table B.1 — Checker rules

185	DTC-DOP	Two objects with the same TROUBLE-CODE must have the same SHORT-NAME and vice versa	error	
186	COMPARAM_REF	The COMPLEX-VALUE at a COMPARAM-REF always must have the same structure as the COMPLEX-COMPARAM it references.	error	
187	IDENT-DESC	Case 1: DIAG-COMM-SNREF refers to a DIAG-SERVICE: The parameter referenced by OUT-PARAM-IF-SNREF must exist in at least one of the DIAG-SERVICE's POS-RESPONSEs. Case 2: DIAG-COMM-SNREF refers to a SINGLE-ECU-JOB: The parameter referenced by OUT-PARAM-IF-SNREF must exist amongst the SINGLE-ECU-JOB's OUTPUT-PARAMs. (Remark: This rule is to be evaluated on the FLASHs.)	error	
188	SESSION-DESC	SHORT-NAMEs of SESSION-DESCs in different ECU-MEM-CONNECTORs referencing the same BASE-VARIANTs or ECU-VARIANTs at LAYER-REFs must be unique in the scope of the LAYER-REFs	Error	
189	STATE-TRANSITION	Within a STATE-TRANSITION it is only allowed to reference STATEs that are contained in the same STATE-CHART as the STATE-TRANSITION itself	Error	
190	STATE-TRANSITION-REF	Only the following IN-PARAM-IF subclasses are allowed within STATE-TRANSITION-REF/STATE-REF: VALUE, CODED-CONST, PHYS-CONST, TABLE-KEY.	Error	
191	STATE-TRANSITION-REF, PRE-CONDITION-STATE-REF	If the IN-PARAM-IF referenced by STATE-TRANSITION-REF or PRE-CONDITION-STATE-REF targets a TABLE-KEY, this TABLE-KEY must have an odx-link to a TABLE-ROW, not a TABLE.	Error	
192	DIAG-LAYER	Only one STATE-CHART per CATERGORY is allowed within one DIAG-LAYER.	Error	
193	STATE-TRANSITION-REF	If STATE-TRANSITION-REF.IN-PARAM-IF is present STATE-TRANSITION-REF.VALUE must be defined.	Error	
194	STATE-TRANSITION	Only DATA-OBJECT-PROPs may be referenced by IN-PARAM-IFs within STATE-TRANSITION/STATE-REF.	Error	
195	ECU-GROUP	Either a PHYS-RESOLUTION-LINK or a FUNCT-RESOLUTION-LINK must exist forevery GROUP-MEMBER. The FUNCT-RESOLUTION-LINK points to a LOGICAL-LINK that references a FUNCTIONAL-GROUP and no BASE-VARIANT. The PHYS-RESOLUTION-LINK points to a LOGICAL-LINK that references a BASE-VARIANT.	Error	
196	LOGICAL-LINK	The TYPE value of the referenced PHYSICAL-LINK-TYPE must exist within the list of POSSIBLE-PHYSICAL-LINK-TYPEs of the used PROTOCOL if defined	Error	
197	VEHICLE-CONNECTOR	Within a VEHICLE-CONNECTOR all PIN_NUMBERs must be unique	Error	
198	PROTOCOL	For every protocol its SHORT-NAME must be CP_UniqueRespIdTable.	Error	
199	ALLOW_MULTIPLE_VALUES	The ALLOW_MULTIPLE_VALUES attribute of the CP_UniqueRespIdTable must be set to “true”.	Error	
200	COMPLEX-	The general content of the COMPLEX-COMPARAM must be filled in		

Table B.1 — Checker rules

	COMPARAM	a protocol-specific manner. The details for standardized protocols can be found in in the D-PDU-API specficiation. For non-standardized protocols this needs to be defined amongst the users of this protocol.		
201	COMPLEX-COMPARAM	The COMPLEX-COMPARAM always shall contain a COMPARAM with SHORT-NAME CP_ECULayerShortName. The CPUSAGE of this communication parameter must be set to APPLICATION.		
202	COMPLEX-COMPARAM	The CPUSAGE of the COMPLEX-COMPARAM and all contained COMPARAMs except the one listed in Rule 200 shall be set to UNIQUE_ID.	Error	
203	LOGICAL-LINK	One of the two link types (PHYS-RESOLUTION-LINK, FUNCT-RESOLUTION-LINK) must be defined for every GROUP-MEMBER. The specification of both kinds of links is equally possible.	Error	
204	LOGICAL-LINK	The FUNCT-RESOLUTION-LINK points to a LOGICAL-LINK that references a FUNCTIONAL-GROUP and no BASE-VARIANT.	Error	
205	LOGICAL-LINK	The PHYS-RESOLUTION-LINK points to a LOGICAL-LINK that references a BASE-VARIANT.	Error	
206	BASE-VARIANT-PATTERN	If the BASE-VARIANT-PATTERN of BASE-VARIANT references at least one MATCHING-PARAMETER with USE-PHYSICAL-ADDRESSING = false, the BASE-VARIANT must have a FUNCTIONAL-RESOLUTION-LINK specified in every ECU-GROUP it appears in.	Error	
207	BASE-VARIANT-PATTERN	If the BASE-VARIANT-PATTERN of BASE-VARIANT references at least one MATCHING-PARAMETER with USE-PHYSICAL-ADDRESSING = true, the BASE-VARIANT must have a PHYS-RESOLUTION-LINK specified in every ECU-GROUP it appears in.	Error	

Annex C
(informative)

Coherent examples

C.1 ISO 14229-1 examples

In this example the following diagnostic services with appropriate DOPs, UNITs etc. are implemented:

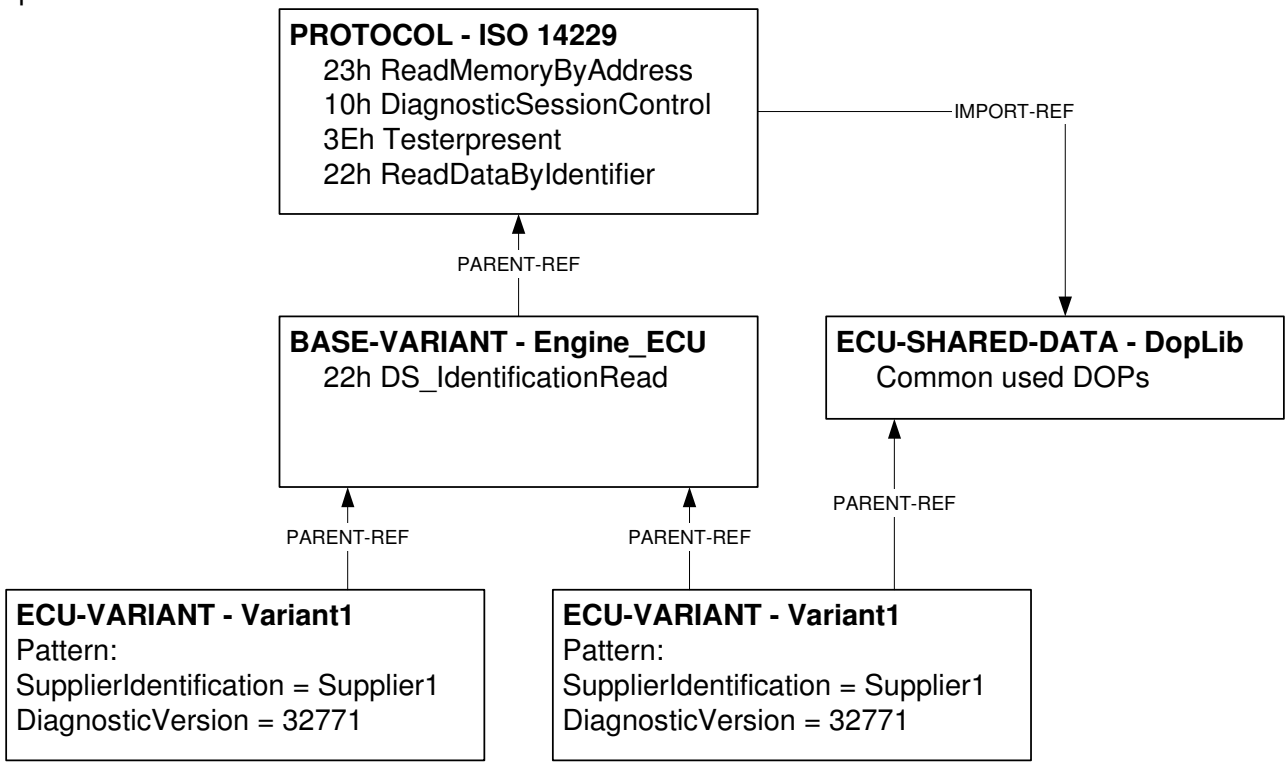


Figure C.1 — ISO 14229-1 example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
       <DIAG-LAYER-CONTAINER ID="ODX_EXAMPLE">
              <SHORT-NAME>ODX_EXAMPLE</SHORT-NAME>
              <LONG-NAME>ODX Example</LONG-NAME>
              <DESC TI="INTRO">
                     <p align="left">This instance contains examples for the usage
of the ODX-Specification. They can not be used with an existing ECU - They are
just for demonstration of the usage of the ODX-XML-Schema</p>
              </DESC>
              <ADMIN-DATA>
                     <LANGUAGE>en_UK</LANGUAGE>
                     <DOC-REVISIONS>
                            <DOC-REVISION>
                                   <TEAM-MEMBER-REF ID-REF="JDo"/>
                                   <REVISION-LABEL>0.1</REVISION-LABEL>
                                   <STATE>INITIAL</STATE>
```

```
                                    <DATE>2004-02-25T15:00:00</DATE>
                                    <TOOL>XMLEDITOR</TOOL>
                                    <MODIFICATIONS>
                                          <MODIFICATION>
                                                <CHANGE>Initial. Addition of Exaple
Services</CHANGE>
                                                <REASON>ODX Documentation</REASON>
                                          </MODIFICATION>
                                    </MODIFICATIONS>
                              </DOC-REVISION>
                              <DOC-REVISION>
                                    <TEAM-MEMBER-REF ID-REF="JSm"/>
                                    <REVISION-LABEL>0.2</REVISION-LABEL>
                                    <STATE>REVIEW</STATE>
                                    <DATE>2003-11-06T15:00:00</DATE>
                                    <MODIFICATIONS>
                                          <MODIFICATION>
                                                <CHANGE>Review/Changes caused by
Review</CHANGE>
                                          </MODIFICATION>
                                    </MODIFICATIONS>
                              </DOC-REVISION>
                        </DOC-REVISIONS>
                  </ADMIN-DATA>
                  <COMPANY-DATAS>
                        <COMPANY-DATA ID="SampleCompany">
                              <SHORT-NAME>SampleCompany</SHORT-NAME>
                              <LONG-NAME>Sample Company</LONG-NAME>
                              <TEAM-MEMBERS>
                                    <TEAM-MEMBER ID="JDo">
                                          <SHORT-NAME>JDo</SHORT-NAME>
                                          <LONG-NAME>John Doe</LONG-NAME>
                                          <DEPARTMENT>Development</DEPARTMENT>
                                          <ADDRESS>5th Avenue</ADDRESS>
                                          <ZIP>55555</ZIP>
                                          <CITY>Big Apple</CITY>
                                          <PHONE>+1 555 1234 567</PHONE>
                                          <EMAIL>John.Doe@sample.com</EMAIL>
                                    </TEAM-MEMBER>
                                    <TEAM-MEMBER ID="JSm">
                                          <SHORT-NAME>JSm</SHORT-NAME>
                                          <LONG-NAME>John Smith</LONG-NAME>
                                          <DEPARTMENT>Service</DEPARTMENT>
                                          <ADDRESS>6th Avenue</ADDRESS>
                                          <ZIP>55555</ZIP>
                                          <CITY>Big Apple</CITY>
                                          <PHONE>+1 555 1234 678</PHONE>
                                          <EMAIL>John.Smith@sample.com</EMAIL>
                                    </TEAM-MEMBER>
                              </TEAM-MEMBERS>
                        </COMPANY-DATA>
                  </COMPANY-DATAS>
                  <PROTOCOLS>
                        <PROTOCOL ID="ISO14229" > <!--
TYPE="ISO_15765_3_on_ISO_15765_2" > -->
                              <SHORT-NAME>ISO14229</SHORT-NAME>
                              <LONG-NAME>ISO14229</LONG-NAME>
                              <FUNCT-CLASSS>
                                    <FUNCT-CLASS
ID="DiagnosticAndCommunicationManagement">
                                          <SHORT-
NAME>DiagnosticAndCommunicationManagement</SHORT-NAME>
                                          <LONG-NAME>Diagnostic and Communication
Management functional unit</LONG-NAME>
                                    </FUNCT-CLASS>
                                    <FUNCT-CLASS ID="DataTransmission">
                                          <SHORT-NAME>DataTransmission</SHORT-NAME>
```

```
                                        <LONG-NAME>Data Transmission functional
unit</LONG-NAME>
                                </FUNCT-CLASS>
                                <FUNCT-CLASS ID="UploadDownload">
                                        <SHORT-NAME>UploadDownload</SHORT-NAME>
                                        <LONG-NAME>Upload Download functional
unit</LONG-NAME>
                                </FUNCT-CLASS>
                        </FUNCT-CLASSS>
                        <DIAG-DATA-DICTIONARY-SPEC>
                                <DATA-OBJECT-PROPS>
                                        <DATA-OBJECT-PROP
ID="DOP_DiagnosticSessionType">
                                                <SHORT-
NAME>DOP_DiagnosticSessionType</SHORT-NAME>
                                                <LONG-
NAME>DiagnosticSessionType</LONG-NAME>
                                                <DESC TI="xx">
                                                        <p align="left">The sub-
function parameter diagnosticSessionType is used by the DiagnosticSessionControl
service to select the specific behavior of the server. Explanations and usage of
the possible diagnostic sessions are detailed below. The following sub-function
values are specified (suppressPosRspMsgIndicationBit (bit 7) not shown)</p>
                                                </DESC>
                                                <COMPU-METHOD>

        <CATEGORY>TEXTTABLE</CATEGORY>
                                                        <COMPU-INTERNAL-TO-PHYS>
                                                                <COMPU-SCALES>
                                                                        <COMPU-SCALE>
                                                                                <SHORT-
LABEL>DS</SHORT-LABEL>
                                                                                <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
                                                                                <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
                                                                                <COMPU-
CONST>

        <VT>DefaultSession</VT>
                                                                                </COMPU-
CONST>
                                                                        </COMPU-SCALE>
                                                                        <COMPU-SCALE>
                                                                                <SHORT-
LABEL>PGRS</SHORT-LABEL>
                                                                                <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
                                                                                <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">2</UPPER-LIMIT>
                                                                                <COMPU-
CONST>

        <VT>ProgrammingSession</VT>
                                                                                </COMPU-
CONST>
                                                                        </COMPU-SCALE>
                                                                        <COMPU-SCALE>
                                                                                <SHORT-
LABEL>EXTDS</SHORT-LABEL>
                                                                                <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">3</LOWER-LIMIT>
                                                                                <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">3</UPPER-LIMIT>
                                                                                <COMPU-
CONST>
```

```
      <VT>ExtendedDiagnosticSession</VT>
                                                                    </COMPU-
CONST>
                                                                  </COMPU-SCALE>
                                                              </COMPU-SCALES>
                                                          </COMPU-INTERNAL-TO-PHYS>
                                                      </COMPU-METHOD>
                                                      <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                          <BIT-LENGTH>7</BIT-LENGTH>
                                                      </DIAG-CODED-TYPE>
                                                      <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UNICODE2STRING"/>
                                                      <INTERNAL-CONSTR>
                                                          <LOWER-LIMIT>0</LOWER-LIMIT>
                                                          <UPPER-LIMIT>127</UPPER-
LIMIT>
                                                          <SCALE-CONSTRS>
                                                              <SCALE-CONSTR
VALIDITY="NOT-VALID">
                                                                  <SHORT-
LABEL>reservedByDocument</SHORT-LABEL>
                                                                  <LOWER-LIMIT
INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
                                                                  <UPPER-LIMIT
INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
                                                              </SCALE-CONSTR>
                                                              <SCALE-CONSTR
VALIDITY="NOT-VALID">
                                                                  <SHORT-
LABEL>reservedByDocument</SHORT-LABEL>
                                                                  <LOWER-LIMIT
INTERVAL-TYPE="CLOSED">4</LOWER-LIMIT>
                                                                  <UPPER-LIMIT
INTERVAL-TYPE="CLOSED">63</UPPER-LIMIT>
                                                              </SCALE-CONSTR>
                                                              <SCALE-CONSTR
VALIDITY="NOT-DEFINED">
                                                                  <SHORT-
LABEL>vehicleManufacturerSpecific</SHORT-LABEL>
                                                                  <LOWER-LIMIT
INTERVAL-TYPE="CLOSED">64</LOWER-LIMIT>
                                                                  <UPPER-LIMIT
INTERVAL-TYPE="CLOSED">95</UPPER-LIMIT>
                                                              </SCALE-CONSTR>
                                                              <SCALE-CONSTR
VALIDITY="NOT-DEFINED">
                                                                  <SHORT-
LABEL>systemSupplierSpecific</SHORT-LABEL>
                                                                  <LOWER-LIMIT
INTERVAL-TYPE="CLOSED">96</LOWER-LIMIT>
                                                                  <UPPER-LIMIT
INTERVAL-TYPE="CLOSED">126</UPPER-LIMIT>
                                                              </SCALE-CONSTR>
                                                              <SCALE-CONSTR
VALIDITY="NOT-VALID">
                                                                  <SHORT-
LABEL>reservedByDocument</SHORT-LABEL>
                                                                  <LOWER-LIMIT
INTERVAL-TYPE="CLOSED">127</LOWER-LIMIT>
                                                                  <UPPER-LIMIT
INTERVAL-TYPE="CLOSED">127</UPPER-LIMIT>
                                                              </SCALE-CONSTR>
                                                          </SCALE-CONSTRS>
                                                      </INTERNAL-CONSTR>
                                                  </DATA-OBJECT-PROP>
```

```
                                        <DATA-OBJECT-PROP ID="DOP_SupplierList">
                                              <SHORT-
NAME>DOP_SupplierList</SHORT-NAME>
                                              <LONG-NAME>SupplierList</LONG-NAME>
                                              <COMPU-METHOD>

      <CATEGORY>TEXTTABLE</CATEGORY>
                                                    <COMPU-INTERNAL-TO-PHYS>
                                                          <COMPU-SCALES>
                                                                <COMPU-SCALE>
                                                                      <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">21</LOWER-LIMIT>
                                                                      <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">21</UPPER-LIMIT>
                                                                      <COMPU-
CONST>

      <VT>Supplier1</VT>
                                                                      </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                      <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">22</LOWER-LIMIT>
                                                                      <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">22</UPPER-LIMIT>
                                                                      <COMPU-
CONST>

      <VT>Supplier2</VT>
                                                                      </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                          </COMPU-SCALES>
                                                    </COMPU-INTERNAL-TO-PHYS>
                                              </COMPU-METHOD>
                                              <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>8</BIT-LENGTH>
                                              </DIAG-CODED-TYPE>
                                              <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UNICODE2STRING"/>
                                        </DATA-OBJECT-PROP>
                                        <DATA-OBJECT-PROP ID="DOP_2ByteHexDump">
                                              <SHORT-
NAME>DOP_2ByteHexDump</SHORT-NAME>
                                              <LONG-NAME>2ByteHexDump</LONG-NAME>
                                              <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                              </COMPU-METHOD>
                                              <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>16</BIT-LENGTH>
                                              </DIAG-CODED-TYPE>
                                              <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32" DISPLAY-RADIX="HEX"/>
                                        </DATA-OBJECT-PROP>
                                        <DATA-OBJECT-PROP
ID="DOP_VarLengthOfMemoryAddress">
                                              <SHORT-
NAME>DOP_VarLengthOfMemoryAddress</SHORT-NAME>
                                              <LONG-
NAME>VarLengthOfMemoryAddress</LONG-NAME>
                                              <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
```

```
                                                </COMPU-METHOD>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="PARAM-LENGTH-INFO-TYPE">
                                                      <LENGTH-KEY-REF ID-
REF="REQ_ReadMemoryByAddress.lengthOfMemoryAddress"/>
                                                </DIAG-CODED-TYPE>
                                                <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                          </DATA-OBJECT-PROP>
                                          <DATA-OBJECT-PROP
ID="DOP_VarLengthOfMemorySize">
                                                <SHORT-
NAME>DOP_VarLengthOfMemorySize</SHORT-NAME>
                                                <LONG-
NAME>VariableLengthOfMemorySize</LONG-NAME>
                                                <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                                </COMPU-METHOD>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="PARAM-LENGTH-INFO-TYPE">
                                                      <LENGTH-KEY-REF ID-
REF="REQ_ReadMemoryByAddress.lengthOfMemorySize"/>
                                                </DIAG-CODED-TYPE>
                                                <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                          </DATA-OBJECT-PROP>
                                    </DATA-OBJECT-PROPS>
                                    <STRUCTURES>
                                          <STRUCTURE ID="STRUCT_RecordIdentifier">
                                                <SHORT-
NAME>STRUCT_RecordIdentifier</SHORT-NAME>
                                                <LONG-NAME>RecordIdentifier</LONG-
NAME>
                                                <PARAMS>
                                                      <PARAM
ID="STRUCT_RecordIdentifier.RecordIdentifier" xsi:type="TABLE-KEY">
                                                            <SHORT-
NAME>RecordIdentifier</SHORT-NAME>
                                                            <LONG-
NAME>RecordIdentifier</LONG-NAME>
                                                            <TABLE-REF ID-
REF="TAB_DataIds"/>
                                                      </PARAM>
                                                </PARAMS>
                                          </STRUCTURE>
                                          <STRUCTURE ID="STRUCT_SupplierList">
                                                <SHORT-
NAME>STRUCT_SupplierList</SHORT-NAME>
                                                <LONG-NAME>SupplierList</LONG-NAME>
                                                <PARAMS>
                                                      <PARAM xsi:type="VALUE">
                                                            <SHORT-
NAME>SupplierList</SHORT-NAME>
                                                            <LONG-
NAME>SupplierList</LONG-NAME>
                                                            <BYTE-
POSITION>0</BYTE-POSITION>
                                                            <DOP-REF ID-
REF="DOP_SupplierList"/>
                                                      </PARAM>
                                                </PARAMS>
                                          </STRUCTURE>
                                          <STRUCTURE ID="STRUCT_DiagnosticVersion">
                                                <SHORT-
NAME>STRUCT_DiagnosticVersion</SHORT-NAME>
```

```
                                                <LONG-NAME>DiagnosticVersion</LONG-
NAME>
                                                <PARAMS>
                                                      <PARAM xsi:type="VALUE">
                                                            <SHORT-
NAME>DiagnosticVersion</SHORT-NAME>
                                                            <LONG-
NAME>DiagnosticVersion</LONG-NAME>
                                                            <BYTE-
POSITION>0</BYTE-POSITION>
                                                            <DOP-REF ID-
REF="DOP_2ByteHexDump"/>
                                                      </PARAM>
                                                </PARAMS>
                                          </STRUCTURE>
                                          <STRUCTURE
ID="STRUCT_RecordIdentifierAndData">
                                                <SHORT-
NAME>STRUCT_RecordIdentifierAndData</SHORT-NAME>
                                                <LONG-
NAME>RecordIdentifierAndData</LONG-NAME>
                                                <PARAMS>
                                                      <PARAM
ID="STRUCT_RecordIdentifierAndData.RecordIdentifier" xsi:type="TABLE-KEY">
                                                            <SHORT-
NAME>RecordIdentifier</SHORT-NAME>
                                                            <LONG-
NAME>RecordIdentifier</LONG-NAME>
                                                            <BYTE-
POSITION>0</BYTE-POSITION>
                                                            <TABLE-REF ID-
REF="TAB_DataIds"/>
                                                      </PARAM>
                                                      <PARAM xsi:type="TABLE-
STRUCT">
                                                            <SHORT-
NAME>Data</SHORT-NAME>
                                                            <LONG-NAME>Data</LONG-
NAME>
                                                            <BYTE-
POSITION>2</BYTE-POSITION>
                                                            <TABLE-KEY-REF ID-
REF="STRUCT_RecordIdentifierAndData.RecordIdentifier"/>
                                                      </PARAM>
                                                </PARAMS>
                                          </STRUCTURE>
                                    </STRUCTURES>
                                    <END-OF-PDU-FIELDS>
                                          <END-OF-PDU-FIELD
ID="EOPDUF_ListOfRecordIdentifiers">
                                                <SHORT-
NAME>EOPDUF_ListOfRecordIdentifiers</SHORT-NAME>
                                                <LONG-
NAME>ListOfRecordIdentifiers</LONG-NAME>
                                                <BASIC-STRUCTURE-REF ID-
REF="STRUCT_RecordIdentifier"/>
                                                <MAX-NUMBER-OF-ITEMS>8</MAX-NUMBER-
OF-ITEMS>
                                          </END-OF-PDU-FIELD>
                                          <END-OF-PDU-FIELD
ID="EOPDUF_ListOfRecordIdentifiersAndData">
                                                <SHORT-
NAME>EOPDUF_ListOfRecordIdentifiersAndData</SHORT-NAME>
                                                <LONG-
NAME>ListOfRecordIdentifiersAndData</LONG-NAME>
                                                <BASIC-STRUCTURE-REF ID-
REF="STRUCT_RecordIdentifierAndData"/>
```

```
                                        </END-OF-PDU-FIELD>
                                    </END-OF-PDU-FIELDS>
                                    <TABLES>
                                        <TABLE ID="TAB_DataIds">
                                            <SHORT-NAME>TAB_DataIds</SHORT-
NAME>
                                            <LONG-NAME>Data identifier</LONG-
NAME>
                                            <KEY-LABEL>Data identifier</KEY-
LABEL>
                                            <STRUCT-LABEL>Structure</STRUCT-
LABEL>
                                            <KEY-DOP-REF ID-
REF="DOP_2ByteIdentical"/>
                                            <TABLE-ROW ID="TABROW_0x0100">
                                                <SHORT-
NAME>TABROW_0x0100</SHORT-NAME>
                                                <LONG-NAME>0x0100</LONG-
NAME>
                                                <KEY>256</KEY>
                                                <STRUCTURE-REF ID-
REF="STRUCT_SupplierList"/>
                                            </TABLE-ROW>
                                            <TABLE-ROW ID="TABROW_0x0101">
                                                <SHORT-
NAME>TABROW_0x0101</SHORT-NAME>
                                                <LONG-NAME>0x0101</LONG-
NAME>
                                                <KEY>257</KEY>
                                                <STRUCTURE-REF ID-
REF="STRUCT_DiagnosticVersion"/>
                                            </TABLE-ROW>
                                        </TABLE>
                                    </TABLES>
                                </DIAG-DATA-DICTIONARY-SPEC>
                                <DIAG-COMMS>
                                    <DIAG-SERVICE ID="DS_ReadMemoryByAddress"
SEMANTIC="MEMORY">
                                        <SHORT-
NAME>DS_ReadMemoryByAddress</SHORT-NAME>
                                        <LONG-NAME>Read Memory By Address</LONG-
NAME>
                                        <DESC TI="DS_ReadMemoryByAddress.DESC">
                                            <p align="left">The
ReadMemoryByAddress service allows the client to request memory data from the
server via provided starting address and size of memory to be read.</p>
                                            <p align="left">Note: It is assumed
that 3-byte memory with common format is used.</p>
                                        </DESC>
                                        <FUNCT-CLASS-REFS>
                                            <FUNCT-CLASS-REF ID-
REF="DataTransmission"/>
                                        </FUNCT-CLASS-REFS>
                                        <AUDIENCE IS-MANUFACTURING="true" IS-
SUPPLIER="true" IS-AFTERSALES="true" IS-DEVELOPMENT="true" IS-
AFTERMARKET="true"/>
                                        <REQUEST-REF ID-
REF="REQ_ReadMemoryByAddress"/>
                                        <POS-RESPONSE-REFS>
                                            <POS-RESPONSE-REF ID-
REF="RESP_ReadMemoryByAddress"/>
                                        </POS-RESPONSE-REFS>
                                    </DIAG-SERVICE>
                                    <DIAG-SERVICE ID="DS_DiagnosticSessionControl"
SEMANTIC="SESSION">
                                        <SHORT-
NAME>DS_DiagnosticSessionControl</SHORT-NAME>
```

```
                                        <LONG-
NAME>DiagnosticSessionControl</LONG-NAME>
                                        <DESC
TI="DS_DiagnosticSessionControl.DESC">
                                              <p align="left">The
DiagnosticSessionControl service is used to enable different diagnostic sessions
in the server(s).</p>
                                              <p align="left">Note: example for
overwriting comparam - see details in documentation</p>
                                        </DESC>
                                        <COMPARAM-REFS>
                                              <COMPARAM-REF ID-REF="COMPARAM-
SPEC.14229.COMPARAM.P2max" DOCREF="14229_CP" DOCTYPE="COMPARAM-SPEC">
                                                    <SIMPLE-VALUE>20</SIMPLE-
VALUE>
                                              </COMPARAM-REF>
                                        </COMPARAM-REFS>

                                        <REQUEST-REF ID-
REF="REQ_DiagnosticSessionControl"/>
                                        <POS-RESPONSE-REFS>
                                              <POS-RESPONSE-REF ID-
REF="RESP_DiagnosticSessionControl"/>
                                        </POS-RESPONSE-REFS>
                                        <POS-RESPONSE-SUPPRESSABLE>
                  <BIT-MASK>80</BIT-MASK>
                  <CODED-CONST-SNREF SHORT-NAME="ParamOfSubFunction"/>
            </POS-RESPONSE-SUPPRESSABLE>
                                  </DIAG-SERVICE>
                                  <DIAG-SERVICE ID="DS_TesterPresent"
SEMANTIC="TESTERPRESENT">
                                        <SHORT-NAME>DS_TesterPresent</SHORT-NAME>
                                        <LONG-NAME>TesterPresent</LONG-NAME>
                                        <DESC TI="DS_TesterPresent.DESC">
                                              <p align="left">This service is
used to indicate to a server (or servers) that a client is still connected to the
vehicle and that certain diagnostic services and/or communication that have been
previously activated are to remain active.
                                              This service is used to keep one or
multiple servers in a diagnostic session other than the defaultSession. This can
either be done by transmitting the TesterPresent request message periodically or
in case of the absence of other diagnostic services to prevent the server(s) from
automatically returning to the defaultSession. The detailed session requirements
that apply to the use of this service when keeping a single server or multiple
servers in a diagnostic session other than the defaultSession can be found in the
implementation specifications of ISO/CD&amp;#160;14229-1.2.<b>IMPORTANT</b>&amp;#160;?
The server and the client shall meet the request and response message behaviour
as specified in section 6.5.2 Request message with subFunction sub-function
parameter and server response behaviour in the event that those addressing
methods are implemented for this service.</p>
                                        </DESC>
                                        <FUNCT-CLASS-REFS>
                                              <FUNCT-CLASS-REF ID-
REF="DiagnosticAndCommunicationManagement"/>
                                        </FUNCT-CLASS-REFS>
                                        <AUDIENCE IS-AFTERMARKET="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-MANUFACTURING="true" IS-
SUPPLIER="true"/>
                                        <REQUEST-REF ID-REF="REQ_TesterPresent"/>
                                  </DIAG-SERVICE>
                                  <DIAG-SERVICE SEMANTIC="MEMORY"
ID="DS_ReadDataByIdentifier">
                                        <SHORT-
NAME>DS_ReadDataByIdentifier</SHORT-NAME>
                                        <LONG-NAME>ReadDataByIdentifier</LONG-
NAME>
                                        <DESC TI="DS_ReadDataByIdentifier.DESC">
```

```
                                                <p align="left">The
ReadDataByIdentifier service allows the client to request data record values from
the server identified by one or more dataIdentifiers.
The client request message contains one or more two (2) byte dataIdentifier
values that identify data record(s) maintained by the server (refer to annex
13.5.5.2.3 for allowed dataIdentifier values). The format and definition of the
dataRecord shall be vehicle manufacturer or system supplier specific, and may
include analog input and output signals, digital input and output signals,
internal data, and system status information if supported by the server.
The server may limit the number of dataIdentifiers that can be simultaneously
requested as agreed upon by the vehicle manufacturer and system supplier.
Upon receiving a ReadDataByIdentifier request, the server shall access the data
elements of the records specified by the dataIdentifier parameter(s) and transmit
their value in one single ReadDataByIdentifier positive response containing the
associated dataRecord parameter(s). The request message may contain the same
dataIdentifier multiple times. The server shall treat each dataIdentifier as a
separate parameter and respond with data for each dataIdentifier as often as
requested.<b>IMPORTANT</b> ? The server and the client shall meet the
request and response message behaviour as specified in section 6.5.3 Request
message without subFunction sub-function parameter and server response behaviour
in the event that those addressing methods are implemented for this service.</p>
                                        </DESC>
                                        <FUNCT-CLASS-REFS>
                                               <FUNCT-CLASS-REF ID-
REF="DataTransmission"/>
                                        </FUNCT-CLASS-REFS>
                                        <AUDIENCE IS-AFTERMARKET="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-MANUFACTURING="true" IS-
SUPPLIER="true"/>
                                        <REQUEST-REF ID-
REF="REQ_ReadDataByIdentifier"/>
                                        <POS-RESPONSE-REFS>
                                               <POS-RESPONSE-REF ID-
REF="RESP_ReadDataByIdentifier"/>
                                        </POS-RESPONSE-REFS>
                                 </DIAG-SERVICE>
                          </DIAG-COMMS>
                          <REQUESTS>
                                 <REQUEST ID="REQ_ReadMemoryByAddress">
                                        <SHORT-
NAME>REQ_ReadMemoryByAddress</SHORT-NAME>
                                        <LONG-
NAME>ReadMemoryByAddressRequest</LONG-NAME>
                                        <PARAMS>
                                               <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                                      <SHORT-NAME>RMBA</SHORT-
NAME>
                                                      <LONG-
NAME>ReadMemoryByAddress</LONG-NAME>
                                                      <BYTE-POSITION>0</BYTE-
POSITION>
                                                      <CODED-VALUE>35</CODED-
VALUE>
                                                      <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                             <BIT-LENGTH>8</BIT-
LENGTH>
                                                      </DIAG-CODED-TYPE>
                                               </PARAM>
                                               <PARAM xsi:type="LENGTH-KEY"
ID="REQ_ReadMemoryByAddress.lengthOfMemoryAddress">
                                                      <SHORT-
NAME>lengthOfMemoryAddress</SHORT-NAME>
                                                      <LONG-
NAME>lengthOfMemoryAddress</LONG-NAME>
```

```
                                                        <BYTE-POSITION>1</BYTE-
POSITION>
                                                        <BIT-POSITION>0</BIT-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_4BitIdentical"/>
                                                  </PARAM>
                                                  <PARAM xsi:type="LENGTH-KEY"
ID="REQ_ReadMemoryByAddress.lengthOfMemorySize">
                                                        <SHORT-
NAME>lengthOfMemorySize</SHORT-NAME>
                                                        <LONG-
NAME>lengthOfMemorySize</LONG-NAME>
                                                        <BYTE-POSITION>1</BYTE-
POSITION>
                                                        <BIT-POSITION>4</BIT-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_4BitIdentical"/>
                                                  </PARAM>
                                                  <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                        <SHORT-NAME>MA</SHORT-NAME>
                                                        <LONG-
NAME>memoryAddress</LONG-NAME>
                                                        <DOP-REF ID-
REF="DOP_VarLengthOfMemoryAddress"/>
                                                  </PARAM>
                                                  <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                        <SHORT-NAME>MS</SHORT-NAME>
                                                        <LONG-NAME>memorySize</LONG-
NAME>
                                                        <DOP-REF ID-
REF="DOP_VarLengthOfMemorySize"/>
                                                  </PARAM>
                                            </PARAMS>
                                      </REQUEST>
                                      <REQUEST ID="REQ_DiagnosticSessionControl">
                                            <SHORT-
NAME>REQ_DiagnosticSessionControl</SHORT-NAME>
                                            <LONG-
NAME>DiagnosticSessionControlRequest</LONG-NAME>
                                            <PARAMS>
                                                  <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                                        <SHORT-NAME>DSC</SHORT-NAME>
                                                        <LONG-
NAME>DiagnosticSessionControl</LONG-NAME>
                                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                                        <CODED-VALUE>16</CODED-
VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                                  <PARAM xsi:type="CODED-CONST">
                                                        <SHORT-
NAME>SuppressPositiveResponse</SHORT-NAME>
                                                        <LONG-
NAME>SuppressPositiveResponse</LONG-NAME>
                                                        <DESC>
                                                              <p>0: do not
suppress</p>
```

```
                                                        <p>1: suppress
positive response</p>
                                                    </DESC>
                                                    <BYTE-POSITION>1</BYTE-
POSITION>
                                                    <BIT-POSITION>7</BIT-
POSITION>
                                                    <CODED-VALUE>0</CODED-VALUE>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>1</BIT-
LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                </PARAM>
                                                <PARAM xsi:type="VALUE"
SEMANTIC="DATA-ID">
                                                    <SHORT-NAME>DS</SHORT-NAME>
                                                    <LONG-
NAME>diagnosticSessionType</LONG-NAME>
                                                    <BYTE-POSITION>1</BYTE-
POSITION>
                                                    <DOP-REF ID-
REF="DOP_DiagnosticSessionType"/>
                                                </PARAM>
                                            </PARAMS>
                                        </REQUEST>
                                        <REQUEST ID="REQ_TesterPresent">
                                            <SHORT-NAME>REQ_TesterPresent</SHORT-
NAME>
                                            <LONG-NAME>TesterPresent Request</LONG-
NAME>
                                            <PARAMS>
                                                <PARAM SEMANTIC="SERVICE-ID"
xsi:type="CODED-CONST">
                                                    <SHORT-NAME>TP</SHORT-NAME>
                                                    <LONG-
NAME>TesterPresent</LONG-NAME>
                                                    <BYTE-POSITION>0</BYTE-
POSITION>
                                                    <CODED-VALUE>62</CODED-
VALUE>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" BASE-TYPE-ENCODING="2C" IS-HIGHLOW-BYTE-ORDER="true"
xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>8</BIT-
LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                </PARAM>
                                                <PARAM xsi:type="CODED-CONST">
                                                    <SHORT-
NAME>SuppressPositiveResponse</SHORT-NAME>
                                                    <LONG-
NAME>SuppressPositiveResponse</LONG-NAME>
                                                    <DESC>
                                                        <p>0: do not
suppress</p>
                                                        <p>1: suppress
positive response</p>
                                                    </DESC>
                                                    <BYTE-POSITION>1</BYTE-
POSITION>
                                                    <BIT-POSITION>7</BIT-
POSITION>
                                                    <CODED-VALUE>1</CODED-VALUE>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
```

```
                                                                <BIT-LENGTH>1</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                </PARAM>
                                                <PARAM xsi:type="CODED-CONST">
                                                        <SHORT-
NAME>ZeroSubfunction</SHORT-NAME>
                                                        <LONG-
NAME>ZeroSubfunction</LONG-NAME>
                                                        <BYTE-POSITION>1</BYTE-
POSITION>
                                                        <BIT-POSITION>0</BIT-
POSITION>
                                                        <CODED-VALUE>0</CODED-VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                                <BIT-LENGTH>7</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                </PARAM>
                                        </PARAMS>
                                </REQUEST>
                                <REQUEST ID="REQ_ReadDataByIdentifier">
                                        <SHORT-
NAME>REQ_ReadDataByIdentifier</SHORT-NAME>
                                        <LONG-NAME>ReadDataByIdentifier
Request</LONG-NAME>
                                        <PARAMS>
                                                <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                                        <SHORT-NAME>RDBI</SHORT-
NAME>
                                                        <LONG-
NAME>ReadDataByIdentifier</LONG-NAME>
                                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                                        <CODED-VALUE>34</CODED-
VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                                <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                </PARAM>
                                                <PARAM xsi:type="VALUE">
                                                        <SHORT-
NAME>RecordDataIdentifiers</SHORT-NAME>
                                                        <LONG-NAME>List of
recordDataIdentifier</LONG-NAME>
                                                        <BYTE-POSITION>1</BYTE-
POSITION>
                                                        <DOP-REF ID-
REF="EOPDUF_ListOfRecordIdentifiers"/>
                                                </PARAM>
                                        </PARAMS>
                                </REQUEST>
                        </REQUESTS>
                        <POS-RESPONSES>
                                <POS-RESPONSE ID="RESP_ReadMemoryByAddress">
                                        <SHORT-
NAME>RESP_ReadMemoryByAddress</SHORT-NAME>
                                        <LONG-
NAME>ReadMemoryByAddressResponse</LONG-NAME>
                                        <PARAMS>
                                                <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERIVCE-ID">
```

```
                                                    <SHORT-NAME>RMBAPR</SHORT-
NAME>
                                                    <LONG-
NAME>ReadMemoryByAddress Response Service Id</LONG-NAME>
                                                    <BYTE-POSITION>0</BYTE-
POSITION>
                                                    <CODED-VALUE>99</CODED-
VALUE>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" IS-HIGHLOW-BYTE-ORDER="true" xsi:type="STANDARD-LENGTH-TYPE">
                                                          <BIT-LENGTH>8</BIT-
LENGTH>
                                                    </DIAG-CODED-TYPE>
                                              </PARAM>
                                              <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                    <SHORT-
NAME>DREC_DATA</SHORT-NAME>
                                                    <LONG-NAME>dataRecord</LONG-
NAME>
                                                    <BYTE-POSITION>1</BYTE-
POSITION>
                                                    <DOP-REF ID-
REF="DOP_EndOfPDUDump"/>
                                              </PARAM>
                                        </PARAMS>
                                  </POS-RESPONSE>
                                  <POS-RESPONSE
ID="RESP_DiagnosticSessionControl">
                                        <SHORT-
NAME>RESP_DiagnosticSessionControl</SHORT-NAME>
                                        <LONG-NAME>DiagnosticSessionControl
Response</LONG-NAME>
                                        <PARAMS>
                                              <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERIVCE-ID">
                                                    <SHORT-NAME>DSCPR</SHORT-
NAME>
                                                    <LONG-
NAME>DiagnosticSessionControl Response Service Id</LONG-NAME>
                                                    <BYTE-POSITION>0</BYTE-
POSITION>
                                                    <CODED-VALUE>80</CODED-
VALUE>
                                                    <DIAG-CODED-TYPE
xsi:type="STANDARD-LENGTH-TYPE" BASE-DATA-TYPE="A_UINT32">
                                                          <BIT-LENGTH>8</BIT-
LENGTH>
                                                    </DIAG-CODED-TYPE>
                                              </PARAM>
                                              <PARAM xsi:type="MATCHING-REQUEST-
PARAM" SEMANTIC="DATA-ID">
                                                    <SHORT-NAME>DS</SHORT-NAME>
                                                    <LONG-
NAME>diagnosticSessionType</LONG-NAME>
                                                    <BYTE-POSITION>1</BYTE-
POSITION>
                                                    <REQUEST-BYTE-
POS>1</REQUEST-BYTE-POS>
                                                    <BYTE-LENGTH>1</BYTE-LENGTH>
                                              </PARAM>
                                              <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                    <SHORT-
NAME>SPREC_DATA</SHORT-NAME>
                                                    <LONG-
NAME>sessionParameterRecord</LONG-NAME>
```

```
                                                        <BYTE-POSITION>2</BYTE-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_EndOfPDUDump"/>
                                                  </PARAM>
                                            </PARAMS>
                                      </POS-RESPONSE>
                                      <POS-RESPONSE ID="RESP_ReadDataByIdentifier">
                                            <SHORT-
NAME>RESP_ReadDataByIdentifier</SHORT-NAME>
                                            <LONG-NAME>ReadDataByIdentifier
Response</LONG-NAME>
                                            <PARAMS>
                                                  <PARAM SEMANTIC="SERVICE-ID"
xsi:type="CODED-CONST">
                                                        <SHORT-NAME>RDBIPR</SHORT-
NAME>
                                                        <LONG-
NAME>ReadDataByIdentifier Response Service Id</LONG-NAME>
                                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                                        <CODED-VALUE>98</CODED-
VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                                  <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                        <SHORT-NAME>data</SHORT-
NAME>
                                                        <LONG-NAME>data</LONG-NAME>
                                                        <BYTE-POSITION>1</BYTE-
POSITION>
                                                        <DOP-REF ID-
REF="EOPDUF_ListOfRecordIdentifiersAndData"/>
                                                  </PARAM>
                                            </PARAMS>
                                      </POS-RESPONSE>
                               </POS-RESPONSES>
                               <IMPORT-REFS>
                                      <IMPORT-REF ID-REF="DopLib"/>
                               </IMPORT-REFS>
                               <COMPARAM-SPEC-REF ID-REF="ISO14229" DOCREF="14229_CP"
DOCTYPE="COMPARAM-SPEC" REVISION="1.0"/>
                        </PROTOCOL>
                  </PROTOCOLS>
                  <ECU-SHARED-DATAS>
                        <ECU-SHARED-DATA ID="DopLib">
                               <SHORT-NAME>DopLib</SHORT-NAME>
                               <LONG-NAME>Library with common used DOPs</LONG-NAME>
                               <DESC TI="DopLib.DESC">
                                      <p align="left">Library containing company wide
accessible DOPs</p>
                               </DESC>
                               <DIAG-DATA-DICTIONARY-SPEC>
                                      <DATA-OBJECT-PROPS>
                                            <DATA-OBJECT-PROP
ID="DOP_1ByteIdentical">
                                                  <SHORT-
NAME>DOP_1ByteIdentical</SHORT-NAME>
                                                  <LONG-NAME>1 byte identical</LONG-
NAME>
                                                  <DESC TI="DOP_1ByteIdentical.DESC">
```

```
                                                    <p>An identical function for
1 byte values</p>
                                                </DESC>
                                                <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                                </COMPU-METHOD>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>8</BIT-LENGTH>
                                                </DIAG-CODED-TYPE>
                                                <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                            </DATA-OBJECT-PROP>
                                            <DATA-OBJECT-PROP
ID="DOP_2ByteIdentical">
                                                <SHORT-
NAME>DOP_2ByteIdentical</SHORT-NAME>
                                                <LONG-NAME>2 byte identical</LONG-
NAME>
                                                <DESC TI="DOP_2ByteIdentical.DESC">
                                                    <p>An identical function for
2 byte values</p>
                                                </DESC>
                                                <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                                </COMPU-METHOD>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>16</BIT-LENGTH>
                                                </DIAG-CODED-TYPE>
                                                <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                            </DATA-OBJECT-PROP>
                                            <DATA-OBJECT-PROP
ID="DOP_3ByteIdentical">
                                                <SHORT-
NAME>DOP_3ByteIdentical</SHORT-NAME>
                                                <LONG-NAME>3ByteIdentical</LONG-
NAME>
                                                <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                                </COMPU-METHOD>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" IS-HIGHLOW-BYTE-ORDER="true" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>24</BIT-LENGTH>
                                                </DIAG-CODED-TYPE>
                                                <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                            </DATA-OBJECT-PROP>
                                            <DATA-OBJECT-PROP ID="DOP_4BitIdentical">
                                                <SHORT-
NAME>DOP_4BitIdentical</SHORT-NAME>
                                                <LONG-NAME>4BitIdentical</LONG-
NAME>
                                                <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                                </COMPU-METHOD>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" IS-HIGHLOW-BYTE-ORDER="true" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>4</BIT-LENGTH>
                                                </DIAG-CODED-TYPE>
                                                <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
```

```
                                        </DATA-OBJECT-PROP>
                                        <DATA-OBJECT-PROP ID="DOP_EndOfPDUDump">
                                               <SHORT-
NAME>DOP_EndOfPDUDump</SHORT-NAME>
                                        <LONG-NAME>EndOfPDUDump</LONG-NAME>
                                        <COMPU-METHOD>

       <CATEGORY>IDENTICAL</CATEGORY>
                                        </COMPU-METHOD>
                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_BYTEFIELD" xsi:type="MIN-MAX-LENGTH-TYPE" TERMINATION="END-OF-PDU">
                                                      <MIN-LENGTH>0</MIN-LENGTH>
                                        </DIAG-CODED-TYPE>
                                        <PHYSICAL-TYPE BASE-DATA-
TYPE="A_BYTEFIELD"/>
                                        </DATA-OBJECT-PROP>
                                 </DATA-OBJECT-PROPS>
                           </DIAG-DATA-DICTIONARY-SPEC>
                     </ECU-SHARED-DATA>
               </ECU-SHARED-DATAS>
               <BASE-VARIANTS>
                     <BASE-VARIANT ID="Engine_ECU">
                           <SHORT-NAME>Engine_ECU</SHORT-NAME>
                           <LONG-NAME>Engine ECU</LONG-NAME>
                           <DESC TI="Engine_ECU.DESC">
                                 <p align="left">This instance contains an
example for the usage of the DDLID related Elements in the ODX-Specification.
They can not be used with an existing ECU - They are just for demonstration of
the usage of the ODX-XML-Schema. Another service is
IOCBI_Desired_Idle_Adjustment_RTD as an example for controlling the ECU.</p>
                           </DESC>
                           <DIAG-COMMS>
                                 <DIAG-SERVICE ID="DS_IdentificationRead"
DIAGNOSTIC-CLASS="VARIANTIDENTIFICATION" SEMANTIC="IDENTIFICATION">
                                        <SHORT-NAME>DS_IdentificationRead</SHORT-
NAME>
                                        <LONG-NAME>InputOutputControlByIdentifier
Desired Idle Adjustment - Reset To Default.</LONG-NAME>
                                        <DESC TI="DIADJ_RTD.DESC">
                                               <p align="left">Example Service
from ISO 14229 Specification Chapter 11.2.5.2</p>
                                        </DESC>
                                        <AUDIENCE IS-AFTERMARKET="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-MANUFACTURING="true" IS-
SUPPLIER="true"/>
                                        <REQUEST-REF ID-
REF="REQ_IdentificationRead"/>
                                        <POS-RESPONSE-REFS>
                                               <POS-RESPONSE-REF ID-
REF="RESP_IdentificationRead"/>
                                        </POS-RESPONSE-REFS>
                                 </DIAG-SERVICE>
                           </DIAG-COMMS>
                           <REQUESTS>
                                 <REQUEST ID="REQ_IdentificationRead">
                                        <SHORT-
NAME>REQ_IdentificationRead</SHORT-NAME>
                                        <LONG-NAME>IdentificationRead
Request</LONG-NAME>
                                        <PARAMS>
                                               <PARAM SEMANTIC="SERVICE-ID"
xsi:type="CODED-CONST">
                                                      <SHORT-
NAME>ServiceId</SHORT-NAME>
                                                      <LONG-
NAME>ReadIdentification</LONG-NAME>
```

```
                                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                                        <CODED-VALUE>34</CODED-
VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                                  <PARAM SEMANTIC="DATA-ID"
xsi:type="CODED-CONST">
                                                        <SHORT-
NAME>SupplierIdentificationId</SHORT-NAME>
                                                        <LONG-
NAME>SupplierIdentificationId</LONG-NAME>
                                                        <BYTE-POSITION>1</BYTE-
POSITION>
                                                        <CODED-VALUE>256</CODED-
VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>16</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                                  <PARAM SEMANTIC="DATA-ID"
xsi:type="CODED-CONST">
                                                        <SHORT-
NAME>DiagnosticVersionId</SHORT-NAME>
                                                        <LONG-
NAME>DiagnosticVersionId</LONG-NAME>
                                                        <BYTE-POSITION>3</BYTE-
POSITION>
                                                        <CODED-VALUE>257</CODED-
VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>16</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                            </PARAMS>
                                      </REQUEST>
                                </REQUESTS>
                                <POS-RESPONSES>
                                      <POS-RESPONSE ID="RESP_IdentificationRead">
                                            <SHORT-
NAME>RESP_IdentificationRead</SHORT-NAME>
                                            <LONG-NAME>IdentificationRead</LONG-NAME>
                                            <PARAMS>
                                                  <PARAM SEMANTIC="SERVICE-ID"
xsi:type="CODED-CONST">
                                                        <SHORT-
NAME>ServiceId</SHORT-NAME>
                                                        <LONG-
NAME>ReadIdentification</LONG-NAME>
                                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                                        <CODED-VALUE>98</CODED-
VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
```

```
                                                <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                      <SHORT-
NAME>SupplierIdentification</SHORT-NAME>
                                                      <LONG-
NAME>SupplierIdentification</LONG-NAME>
                                                      <BYTE-POSITION>3</BYTE-
POSITION>
                                                      <DOP-REF ID-
REF="DOP_SupplierList"/>
                                                </PARAM>
                                                <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                      <SHORT-
NAME>DiagnosticVersion</SHORT-NAME>
                                                      <LONG-
NAME>DiagnosticVersion</LONG-NAME>
                                                      <BYTE-POSITION>6</BYTE-
POSITION>
                                                      <DOP-REF ID-
REF="DOP_2ByteHexDump"/>
                                                </PARAM>
                                          </PARAMS>
                                    </POS-RESPONSE>
                              </POS-RESPONSES>
                              <PARENT-REFS>
                                    <PARENT-REF ID-REF="ISO14229"
xsi:type="PROTOCOL-REF"/>
                                    <PARENT-REF ID-REF="DopLib" xsi:type="ECU-
SHARED-DATA-REF"/>
                              </PARENT-REFS>
                        </BASE-VARIANT>
                  </BASE-VARIANTS>
                  <ECU-VARIANTS>
                        <ECU-VARIANT ID="Variant1">
                              <SHORT-NAME>Variant1</SHORT-NAME>
                              <LONG-NAME>Engine - Variant 1</LONG-NAME>
                              <ECU-VARIANT-PATTERNS>
                                    <ECU-VARIANT-PATTERN>
                                          <MATCHING-PARAMETERS>
                                                <MATCHING-PARAMETER>
                                                      <EXPECTED-
VALUE>Supplier1</EXPECTED-VALUE>
                                                      <DIAG-COMM-SNREF SHORT-
NAME="DS_IdentificationRead"/>
                                                      <OUT-PARAM-IF-SNREF SHORT-
NAME="SupplierIdentification"/>
                                                </MATCHING-PARAMETER>
                                                <MATCHING-PARAMETER>
                                                      <EXPECTED-
VALUE>32771</EXPECTED-VALUE>
                                                      <DIAG-COMM-SNREF SHORT-
NAME="DS_IdentificationRead"/>
                                                      <OUT-PARAM-IF-SNREF SHORT-
NAME="DiagnosticVersion"/>
                                                </MATCHING-PARAMETER>
                                          </MATCHING-PARAMETERS>
                                    </ECU-VARIANT-PATTERN>
                              </ECU-VARIANT-PATTERNS>
                              <PARENT-REFS>
                                    <PARENT-REF ID-REF="Engine_ECU" xsi:type="BASE-
VARIANT-REF">
                                          <NOT-INHERITED-DIAG-COMMS>
                                                <NOT-INHERITED-DIAG-COMM>
                                                      <DIAG-COMM-SNREF SHORT-
NAME="DS_ReadMemoryByAddress"/>
                                                </NOT-INHERITED-DIAG-COMM>
```

```
                                        </NOT-INHERITED-DIAG-COMMS>
                                  </PARENT-REF>
                            </PARENT-REFS>
                      </ECU-VARIANT>
                      <ECU-VARIANT ID="Variant2">
                            <SHORT-NAME>Variant2</SHORT-NAME>
                            <LONG-NAME>Engine - Variant 2</LONG-NAME>
                            <ECU-VARIANT-PATTERNS>
                                  <ECU-VARIANT-PATTERN>
                                        <MATCHING-PARAMETERS>
                                              <MATCHING-PARAMETER>
                                                    <EXPECTED-
VALUE>Supplier2</EXPECTED-VALUE>
                                                    <DIAG-COMM-SNREF SHORT-
NAME="DS_IdentificationRead"/>
                                                    <OUT-PARAM-IF-SNREF SHORT-
NAME="SupplierIdentification"/>
                                              </MATCHING-PARAMETER>
                                              <MATCHING-PARAMETER>
                                                    <EXPECTED-
VALUE>32771</EXPECTED-VALUE>
                                                    <DIAG-COMM-SNREF SHORT-
NAME="DS_IdentificationRead"/>
                                                    <OUT-PARAM-IF-SNREF SHORT-
NAME="DiagnosticVersion"/>
                                              </MATCHING-PARAMETER>
                                        </MATCHING-PARAMETERS>
                                  </ECU-VARIANT-PATTERN>
                            </ECU-VARIANT-PATTERNS>
                            <PARENT-REFS>
                                  <PARENT-REF ID-REF="Engine_ECU" xsi:type="BASE-
VARIANT-REF"/>
                            </PARENT-REFS>
                      </ECU-VARIANT>
                </ECU-VARIANTS>
          </DIAG-LAYER-CONTAINER>
</ODX>
```

C.2 ISO 14230 examples

This XML code contains five services of ISO 14230 used for reading of DTCs, defining of DDLID and reading of DDLID records. These services are:

- ReadStatusOfDTC (ServiceID=17h)
- ReadDTCByStatus (ServiceID=18h)
- DDLID_READ (ReadDataByLID, ServiceID=21h)
- DDLID_STATIC (DynamicallyDefineLocalID, ServiceID=2Ch)
- DDLID_DYNAMIC (DynamicallyDefineLocalID, ServiceID=2Ch)

```
<?xml version="1.0" encoding="UTF-8"?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
      <DIAG-LAYER-CONTAINER ID="ISO14230Example">
            <SHORT-NAME>ISO14230Example</SHORT-NAME>
            <LONG-NAME>ISO14230 Example for the ODX Specification</LONG-NAME>
            <ADMIN-DATA>
```

```
                    <LANGUAGE>de_DE</LANGUAGE>
                    <DOC-REVISIONS>
                          <DOC-REVISION>
                                <TEAM-MEMBER-REF ID-REF="JDo"/>
                                <REVISION-LABEL>0.1</REVISION-LABEL>
                                <STATE>INITIAL</STATE>
                                <DATE>2004-02-25T00:00:00</DATE>
                                <TOOL>XMLSPY</TOOL>
                                <MODIFICATIONS>
                                      <MODIFICATION>
                                            <CHANGE>Initial. Addition of Exaple
Services</CHANGE>
                                            <REASON>ODX Documentation</REASON>
                                      </MODIFICATION>
                                </MODIFICATIONS>
                          </DOC-REVISION>
                    </DOC-REVISIONS>
              </ADMIN-DATA>
              <COMPANY-DATAS>
                    <COMPANY-DATA ID="SampleCompany">
                          <SHORT-NAME>SampleCompany</SHORT-NAME>
                          <LONG-NAME>Sample Company</LONG-NAME>
                          <TEAM-MEMBERS>
                                <TEAM-MEMBER ID="JDo">
                                      <SHORT-NAME>JDo</SHORT-NAME>
                                      <LONG-NAME>John Doe</LONG-NAME>
                                      <ROLES>
                                            <ROLE>Author</ROLE>
                                      </ROLES>
                                      <DEPARTMENT>Development</DEPARTMENT>
                                      <ADDRESS>5th Avenue</ADDRESS>
                                      <ZIP>55555</ZIP>
                                      <CITY>Big Apple</CITY>
                                      <PHONE>+1 555 1234 567</PHONE>
                                      <FAX>+1 555 1234 999</FAX>
                                      <EMAIL>John.Doe@sample.com</EMAIL>
                                </TEAM-MEMBER>
                                <TEAM-MEMBER ID="JSm">
                                      <SHORT-NAME>JSm</SHORT-NAME>
                                      <LONG-NAME>John Smith</LONG-NAME>
                                      <ROLES>
                                            <ROLE>Author</ROLE>
                                      </ROLES>
                                      <DEPARTMENT>Service</DEPARTMENT>
                                      <ADDRESS>6th Avenue</ADDRESS>
                                      <ZIP>55    555</ZIP>
                                      <CITY>Big Apple</CITY>
                                      <PHONE>+1 555 1234 678</PHONE>
                                      <FAX>+1 555 1234 888</FAX>
                                      <EMAIL>John.Smith@sample.com</EMAIL>
                                </TEAM-MEMBER>
                          </TEAM-MEMBERS>
                    </COMPANY-DATA>
              </COMPANY-DATAS>
              <PROTOCOLS>
                    <PROTOCOL ID="ISO14230" TYPE="ISO_14230_3_on_ISO_14230_2">
                          <SHORT-NAME>ISO14230</SHORT-NAME>
                          <LONG-NAME>ISO14230</LONG-NAME>
                          <ADMIN-DATA>
                                <DOC-REVISIONS>
                                      <DOC-REVISION>
                                            <DATE>2004-02-25T00:00:00</DATE>
                                            <TOOL>XMLSPY 2004</TOOL>
                                            <MODIFICATIONS>
                                                  <MODIFICATION>
                                                        <CHANGE>added DTC-
DOP</CHANGE>
```

```
      <REASON>Example</REASON>
                                                </MODIFICATION>
                                          </MODIFICATIONS>
                                    </DOC-REVISION>
                              </DOC-REVISIONS>
                        </ADMIN-DATA>
                        <DIAG-DATA-DICTIONARY-SPEC>
                              <DTC-DOPS>
                                    <DTC-DOP ID="DTCDOP_AllDTCs">
                                          <SHORT-NAME>DTCDOP_AllDTCs</SHORT-
NAME>
                                          <LONG-NAME>All DTCs</LONG-NAME>
                                          <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                <BIT-LENGTH>8</BIT-LENGTH>
                                          </DIAG-CODED-TYPE>
                                          <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                          <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                          </COMPU-METHOD>
                                          <DTCS>
                                                <DTC ID="DTC_0120">
                                                      <SHORT-
NAME>DTC_0120</SHORT-NAME>
                                                      <TROUBLE-
CODE>288</TROUBLE-CODE>
                                                      <TEXT>DTC 0120h
Description</TEXT>
                                                      <LEVEL>4</LEVEL>
                                                </DTC>
                                                <DTC ID="DTC_0130">
                                                      <SHORT-
NAME>DTC_0130</SHORT-NAME>
                                                      <TROUBLE-
CODE>304</TROUBLE-CODE>
                                                      <TEXT>DTC 0130h
Description</TEXT>
                                                      <LEVEL>2</LEVEL>
                                                </DTC>
                                          </DTCS>
                                    </DTC-DOP>
                              </DTC-DOPS>
                              <ENV-DATA-DESCS>
                                    <ENV-DATA-DESC ID="ENVDESC_EnvDataDesc">
                                          <SHORT-
NAME>ENVDESC_EnvDataDesc</SHORT-NAME>
                                          <LONG-NAME>Environmental
Data</LONG-NAME>
                                          <PARAM-SNREF SHORT-
NAME="SwitchKeyDTC"/>
                                          <ENV-DATAS>
                                                <ENV-DATA ID="ENVDATA_0120">
                                                      <SHORT-
NAME>ENVDATA_0120</SHORT-NAME>
                                                      <LONG-NAME>ENVDATA
0120h</LONG-NAME>
                                                      <PARAMS>
                                                            <PARAM
xsi:type="VALUE" SEMANTIC="ENVDATA">
                                                                  <SHORT-
NAME>EngineSpeed</SHORT-NAME>
                                                                  <LONG-
NAME>Engine Speed</LONG-NAME>
```

```
                                                                    <BYTE-
POSITION>3</BYTE-POSITION>
                                                                    <DOP-REF
ID-REF="DOP_EngineSpeed"/>
                                                                </PARAM>
                                                                <PARAM
xsi:type="VALUE" SEMANTIC="ENVDATA">
                                                                    <SHORT-
NAME>VehicleSpeed</SHORT-NAME>
                                                                    <LONG-
NAME>Vehicle Speed</LONG-NAME>
                                                                    <BYTE-
POSITION>5</BYTE-POSITION>
                                                                    <DOP-REF
ID-REF="DOP_VehicleSpeed"/>
                                                                </PARAM>
                                                                <PARAM
xsi:type="VALUE" SEMANTIC="ENVDATA">
                                                                    <SHORT-
NAME>EventCount</SHORT-NAME>
                                                                    <LONG-
NAME>Event Counter</LONG-NAME>
                                                                    <BYTE-
POSITION>6</BYTE-POSITION>
                                                                    <DOP-REF
ID-REF="DOP_1ByteIdentical"/>
                                                                </PARAM>
                                                            </PARAMS>
                                                            <DTC-VALUES>
                                                                <DTC-
VALUE>288</DTC-VALUE>
                                                            </DTC-VALUES>
                                                        </ENV-DATA>
                                                        <ENV-DATA ID="ENVDATA_0130">
                                                            <SHORT-
NAME>ENVDATA_0130</SHORT-NAME>
                                                            <PARAMS>
                                                                <PARAM
xsi:type="VALUE" SEMANTIC="ENVDATA">
                                                                    <SHORT-
NAME>Voltage</SHORT-NAME>
                                                                    <LONG-
NAME>Voltage</LONG-NAME>
                                                                    <BYTE-
POSITION>3</BYTE-POSITION>
                                                                    <DOP-REF
ID-REF="DOP_Voltage"/>
                                                                </PARAM>
                                                            </PARAMS>
                                                            <DTC-VALUES>
                                                                <DTC-
VALUE>304</DTC-VALUE>
                                                            </DTC-VALUES>
                                                        </ENV-DATA>
                                                        <ENV-DATA ID="ENVDATA_All">
                                                            <SHORT-
NAME>ENVDATA_All</SHORT-NAME>
                                                            <LONG-NAME>ENVDATA
ALL</LONG-NAME>
                                                            <PARAMS>
                                                                <PARAM
xsi:type="VALUE" SEMANTIC="ENVDATA">
                                                                    <SHORT-
NAME>CurrentMileage</SHORT-NAME>
                                                                    <LONG-
NAME>Current Mileage</LONG-NAME>
```

```
                                                                    <BYTE-
POSITION>0</BYTE-POSITION>
                                                                    <DOP-REF
ID-REF="DOP_CurrentMileage"/>
                                                                </PARAM>
                                                            </PARAMS>
                                                            <ALL-VALUE/>
                                                        </ENV-DATA>
                                                    </ENV-DATAS>
                                                </ENV-DATA-DESC>
                                            </ENV-DATA-DESCS>
                                            <DATA-OBJECT-PROPS>
                                                <DATA-OBJECT-PROP
ID="DOP_CurrentMileage">
                                                    <SHORT-
NAME>DOP_CurrentMileage</SHORT-NAME>
                                                    <LONG-NAME>Current Mileage</LONG-
NAME>
                                                    <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                                    </COMPU-METHOD>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>24</BIT-LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                    <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                                    <UNIT-REF ID-REF="Km"/>
                                                </DATA-OBJECT-PROP>
                                                <DATA-OBJECT-PROP ID="DOP_Voltage">
                                                    <SHORT-NAME>DOP_Voltage</SHORT-
NAME>
                                                    <LONG-NAME>Voltage</LONG-NAME>
                                                    <COMPU-METHOD>
                                                        <CATEGORY>LINEAR</CATEGORY>
                                                        <COMPU-INTERNAL-TO-PHYS>
                                                            <COMPU-SCALES>
                                                                <COMPU-SCALE>
                                                                    <LOWER-
LIMIT>0</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>255</UPPER-LIMIT>
                                                                    <COMPU-
RATIONAL-COEFFS>

      <COMPU-NUMERATOR>

      <V>0</V>

      <V>10</V>

      </COMPU-NUMERATOR>
                                                                    </COMPU-
RATIONAL-COEFFS>
                                                                </COMPU-SCALE>
                                                            </COMPU-SCALES>
                                                        </COMPU-INTERNAL-TO-PHYS>
                                                    </COMPU-METHOD>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>8</BIT-LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                    <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                                    <UNIT-REF ID-REF="Volt"/>
                                                </DATA-OBJECT-PROP>
```

```
                                        <DATA-OBJECT-PROP
ID="DOP_EngineCoolantTemperature">
                                            <SHORT-
NAME>DOP_EngineCoolantTemperature</SHORT-NAME>
                                            <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                            </COMPU-METHOD>
                                            <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                <BIT-LENGTH>8</BIT-LENGTH>
                                            </DIAG-CODED-TYPE>
                                            <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                            <UNIT-REF ID-REF="DegreeCelsius"/>
                                        </DATA-OBJECT-PROP>
                                        <DATA-OBJECT-PROP ID="DOP_VehicleSpeed">
                                            <SHORT-
NAME>DOP_VehicleSpeed</SHORT-NAME>
                                            <LONG-NAME>Vehicle Speed</LONG-
NAME>
                                            <COMPU-METHOD>
                                                <CATEGORY>LINEAR</CATEGORY>
                                                <COMPU-INTERNAL-TO-PHYS>
                                                    <COMPU-SCALES>
                                                        <COMPU-SCALE>
                                                            <LOWER-
LIMIT>0</LOWER-LIMIT>
                                                            <UPPER-
LIMIT>255</UPPER-LIMIT>
                                                            <COMPU-
RATIONAL-COEFFS>

      <COMPU-NUMERATOR>

      <V>0</V>

      <V>1.07143</V>

      </COMPU-NUMERATOR>
                                                            </COMPU-
RATIONAL-COEFFS>
                                                        </COMPU-SCALE>
                                                    </COMPU-SCALES>
                                                </COMPU-INTERNAL-TO-PHYS>
                                            </COMPU-METHOD>
                                            <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                <BIT-LENGTH>8</BIT-LENGTH>
                                            </DIAG-CODED-TYPE>
                                            <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                            <UNIT-REF ID-REF="Kmh"/>
                                        </DATA-OBJECT-PROP>
                                        <DATA-OBJECT-PROP ID="DOP_EngineSpeed">
                                            <SHORT-NAME>DOP_EngineSpeed</SHORT-
NAME>
                                            <LONG-NAME>Engine Speed</LONG-NAME>
                                            <COMPU-METHOD>
                                                <CATEGORY>LINEAR</CATEGORY>
                                                <COMPU-INTERNAL-TO-PHYS>
                                                    <COMPU-SCALES>
                                                        <COMPU-SCALE>
                                                            <LOWER-
LIMIT>0</LOWER-LIMIT>
                                                            <UPPER-
LIMIT>255</UPPER-LIMIT>
```

```
                                                                    <COMPU-
RATIONAL-COEFFS>
       <COMPU-NUMERATOR>
       <V>0</V>
       <V>1</V>
       </COMPU-NUMERATOR>
       <COMPU-DENOMINATOR>
       <V>4</V>
       </COMPU-DENOMINATOR>
                                                                    </COMPU-
RATIONAL-COEFFS>
                                                                 </COMPU-SCALE>
                                                              </COMPU-SCALES>
                                                           </COMPU-INTERNAL-TO-PHYS>
                                                        </COMPU-METHOD>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_FLOAT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                           <BIT-LENGTH>16</BIT-LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                        <PHYSICAL-TYPE BASE-DATA-
TYPE="A_FLOAT32">
                                                           <PRECISION>2</PRECISION>
                                                        </PHYSICAL-TYPE>
                                                        <UNIT-REF ID-REF="Rpm"/>
                                                     </DATA-OBJECT-PROP>
                                                     <DATA-OBJECT-PROP
ID="DOP_ThrottlePosition">
                                                        <SHORT-
NAME>DOP_ThrottlePosition</SHORT-NAME>
                                                        <LONG-NAME>Throttle Position</LONG-
NAME>
                                                        <COMPU-METHOD>
       <CATEGORY>IDENTICAL</CATEGORY>
                                                        </COMPU-METHOD>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                           <BIT-LENGTH>16</BIT-LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                        <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                                        <UNIT-REF ID-REF="Rpm"/>
                                                     </DATA-OBJECT-PROP>
                                                     <DATA-OBJECT-PROP
ID="DOP_WarningLampCalibrationStatus">
                                                        <SHORT-
NAME>DOP_WarningLampCalibrationStatus</SHORT-NAME>
                                                        <LONG-NAME>DTC Warning Lamp
Calibration Status</LONG-NAME>
                                                        <COMPU-METHOD>
       <CATEGORY>TEXTTABLE</CATEGORY>
                                                           <COMPU-INTERNAL-TO-PHYS>
                                                              <COMPU-SCALES>
                                                                 <COMPU-SCALE>
                                                                    <SHORT-
LABEL>WLD</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
```

```
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
                                                                    <COMPU-
CONST>

       <VT>warning lamp disabled</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>WLE</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
                                                                    <COMPU-
CONST>

       <VT>warning lamp enabled</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                            </COMPU-SCALES>
                                                        </COMPU-INTERNAL-TO-PHYS>
                                                    </COMPU-METHOD>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>1</BIT-LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                    <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UNICODE2STRING"/>
                                                    <INTERNAL-CONSTR>
                                                        <LOWER-LIMIT INTERVAL-
TYPE="CLOSED">0</LOWER-LIMIT>
                                                        <UPPER-LIMIT INTERVAL-
TYPE="CLOSED">1</UPPER-LIMIT>
                                                    </INTERNAL-CONSTR>
                                                </DATA-OBJECT-PROP>
                                                <DATA-OBJECT-PROP ID="DOP_StorageStatus">
                                                    <SHORT-
NAME>DOP_StorageStatus</SHORT-NAME>
                                                    <LONG-NAME>DTC Storage
Status</LONG-NAME>
                                                    <COMPU-METHOD>

       <CATEGORY>TEXTTABLE</CATEGORY>
                                                        <COMPU-INTERNAL-TO-PHYS>
                                                            <COMPU-SCALES>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>NODTC</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
                                                                    <COMPU-
CONST>

       <VT>no DTC present at time of request</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>DTCNP</SHORT-LABEL>
```

```
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
       <VT>DTC not present at time of request</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>DTCMI</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">2</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
       <VT>DTC maturing-intermittent at time of request</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>DTCP</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">3</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">3</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
       <VT>DTC present at time of request</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                            </COMPU-SCALES>
                                                        </COMPU-INTERNAL-TO-PHYS>
                                                    </COMPU-METHOD>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>2</BIT-LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                    <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UNICODE2STRING"/>
                                                    <INTERNAL-CONSTR>
                                                        <LOWER-LIMIT INTERVAL-
TYPE="CLOSED">0</LOWER-LIMIT>
                                                        <UPPER-LIMIT INTERVAL-
TYPE="CLOSED">3</UPPER-LIMIT>
                                                    </INTERNAL-CONSTR>
                                                </DATA-OBJECT-PROP>
                                                <DATA-OBJECT-PROP ID="DOP_ReadynessFlag">
                                                    <SHORT-
NAME>DOP_ReadynessFlag</SHORT-NAME>
                                                    <LONG-NAME>DTC Readyness
Flag</LONG-NAME>
                                                    <COMPU-METHOD>
       <CATEGORY>TEXTTABLE</CATEGORY>
                                                        <COMPU-INTERNAL-TO-PHYS>
                                                            <COMPU-SCALES>
                                                                <COMPU-SCALE>
```

```
                                                                    <SHORT-
LABEL>TC</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
       <VT>test complete for this DTC</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>TNC</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
       <VT>test not complete for this DTC</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                            </COMPU-SCALES>
                                                        </COMPU-INTERNAL-TO-PHYS>
                                                    </COMPU-METHOD>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>1</BIT-LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                    <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UNICODE2STRING"/>
                                                    <INTERNAL-CONSTR>
                                                        <LOWER-LIMIT INTERVAL-
TYPE="CLOSED">0</LOWER-LIMIT>
                                                        <UPPER-LIMIT INTERVAL-
TYPE="CLOSED">1</UPPER-LIMIT>
                                                    </INTERNAL-CONSTR>
                                                </DATA-OBJECT-PROP>
                                                <DATA-OBJECT-PROP ID="DOP_FaultSymptom">
                                                    <SHORT-
NAME>DOP_FaultSymptom</SHORT-NAME>
                                                    <LONG-NAME>DTC Fault Symptom</LONG-
NAME>
                                                    <COMPU-METHOD>
       <CATEGORY>TEXTTABLE</CATEGORY>
                                                        <COMPU-INTERNAL-TO-PHYS>
                                                            <COMPU-SCALES>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>NFS</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
       <VT>no fault symptom</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
```

```
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>AMT</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>above maximum threshold</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>BMT</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">2</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>below minimum threshold</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>NS</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">4</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">4</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>no signal</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>IS</SHORT-LABEL>
                                                                    <LOWER-
LIMIT INTERVAL-TYPE="CLOSED">8</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT INTERVAL-TYPE="CLOSED">8</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>invalid signal</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                            </COMPU-SCALES>
                                                            <COMPU-DEFAULT-VALUE>
                                                                <VT>not
defined</VT>
                                                            </COMPU-DEFAULT-VALUE>
                                                        </COMPU-INTERNAL-TO-PHYS>
                                                    </COMPU-METHOD>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>4</BIT-LENGTH>
                                                    </DIAG-CODED-TYPE>
```

```
                                                <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UNICODE2STRING"/>
                                                <INTERNAL-CONSTR>
                                                      <LOWER-LIMIT INTERVAL-
TYPE="CLOSED">0</LOWER-LIMIT>
                                                      <UPPER-LIMIT INTERVAL-
TYPE="CLOSED">15</UPPER-LIMIT>
                                                </INTERNAL-CONSTR>
                                          </DATA-OBJECT-PROP>
                                          <DATA-OBJECT-PROP
ID="DOP_1ByteIdentical">
                                                <SHORT-
NAME>DOP_1ByteIdentical</SHORT-NAME>
                                                <LONG-NAME>1 Byte Identical</LONG-
NAME>
                                                <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                                </COMPU-METHOD>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                      <BIT-LENGTH>8</BIT-LENGTH>
                                                </DIAG-CODED-TYPE>
                                                <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                          </DATA-OBJECT-PROP>
                                          <DATA-OBJECT-PROP
ID="DOP_2ByteIdentical">
                                                <SHORT-
NAME>DOP_2ByteIdentical</SHORT-NAME>
                                                <LONG-NAME>2 Bytes Identical</LONG-
NAME>
                                                <COMPU-METHOD>

      <CATEGORY>IDENTICAL</CATEGORY>
                                                </COMPU-METHOD>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                      <BIT-LENGTH>16</BIT-LENGTH>
                                                </DIAG-CODED-TYPE>
                                                <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UINT32"/>
                                          </DATA-OBJECT-PROP>
                                          <DATA-OBJECT-PROP ID="DOP_GroupOfDTC">
                                                <SHORT-NAME>DOP_GroupOfDTC</SHORT-
NAME>
                                                <LONG-NAME>group of DTC</LONG-NAME>
                                                <COMPU-METHOD>

      <CATEGORY>TEXTTABLE</CATEGORY>
                                                      <COMPU-INTERNAL-TO-PHYS>
                                                            <COMPU-SCALES>
                                                                  <COMPU-SCALE>
                                                                        <LOWER-
LIMIT>0</LOWER-LIMIT>
                                                                        <UPPER-
LIMIT>0</UPPER-LIMIT>
                                                                        <COMPU-
CONST>

      <VT>Powertrain Group</VT>
                                                                        </COMPU-
CONST>
                                                                  </COMPU-SCALE>
                                                                  <COMPU-SCALE>
                                                                        <LOWER-
LIMIT>1</LOWER-LIMIT>
```

```
                                                                    <UPPER-
LIMIT>16383</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>Powertrain DTCs</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <LOWER-
LIMIT>16384</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>16384</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>Chassis Group</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <LOWER-
LIMIT>16385</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>32767</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>Chassis DTCs</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <LOWER-
LIMIT>32768</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>32768</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>Body Group</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <LOWER-
LIMIT>32769</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>49151</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>Body DTCs</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <LOWER-
LIMIT>49152</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>49152</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>Network Group</VT>
```

```
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <LOWER-
LIMIT>49153</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>65279</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>Network DTCs</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <LOWER-
LIMIT>65280</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>65280</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>All DTCs</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                            </COMPU-SCALES>
                                                        </COMPU-INTERNAL-TO-PHYS>
                                                    </COMPU-METHOD>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>16</BIT-LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                    <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UNICODE2STRING"/>
                                                </DATA-OBJECT-PROP>
                                                <DATA-OBJECT-PROP
ID="DOP_StatusOfDTCReq">
                                                    <SHORT-
NAME>DOP_StatusOfDTCReq</SHORT-NAME>
                                                    <LONG-NAME>status of DTC in
request</LONG-NAME>
                                                    <COMPU-METHOD>
      <CATEGORY>TEXTTABLE</CATEGORY>
                                                        <COMPU-INTERNAL-TO-PHYS>
                                                            <COMPU-SCALES>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>RIDTCAS</SHORT-LABEL>
                                                                    <LOWER-
LIMIT>0</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>0</UPPER-LIMIT>
                                                                    <COMPU-
CONST>
      <VT>request Identified BCD DTC and Status</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>RSUDTCAS</SHORT-LABEL>
                                                                    <LOWER-
LIMIT>1</LOWER-LIMIT>
```

```
                                                                    <UPPER-
LIMIT>1</UPPER-LIMIT>
                                                                    <COMPU-
CONST>

       <VT>request Supported DTC and Status</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                                <COMPU-SCALE>
                                                                    <SHORT-
LABEL>RI2BHDTCAS</SHORT-LABEL>
                                                                    <LOWER-
LIMIT>2</LOWER-LIMIT>
                                                                    <UPPER-
LIMIT>2</UPPER-LIMIT>
                                                                    <COMPU-
CONST>

       <VT>request identified 2 Byte Hex DTC and Status</VT>
                                                                    </COMPU-
CONST>
                                                                </COMPU-SCALE>
                                                            </COMPU-SCALES>
                                                            <COMPU-DEFAULT-VALUE>
                                                                <VT>not
supported</VT>
                                                            </COMPU-DEFAULT-VALUE>
                                                        </COMPU-INTERNAL-TO-PHYS>
                                                    </COMPU-METHOD>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>8</BIT-LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                    <PHYSICAL-TYPE BASE-DATA-
TYPE="A_UNICODE2STRING"/>
                                            </DATA-OBJECT-PROP>
                                        </DATA-OBJECT-PROPS>
                                        <STRUCTURES>
                                            <STRUCTURE ID="STRUCT_StatusOfDTC">
                                                <SHORT-
NAME>STRUCT_StatusOfDTC</SHORT-NAME>
                                                <LONG-NAME>Status of DTC</LONG-
NAME>
                                                <BYTE-SIZE>1</BYTE-SIZE>
                                                <PARAMS>
                                                    <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                        <SHORT-
NAME>DTCWLCS</SHORT-NAME>
                                                        <LONG-NAME>DTC Warning
Lamp Calibration Status</LONG-NAME>
                                                        <BYTE-
POSITION>0</BYTE-POSITION>
                                                        <BIT-POSITION>7</BIT-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_WarningLampCalibrationStatus"/>
                                                    </PARAM>
                                                    <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                        <SHORT-
NAME>DTCSS</SHORT-NAME>
                                                        <LONG-NAME>DTC Storage
Status</LONG-NAME>
                                                        <BYTE-
POSITION>0</BYTE-POSITION>
```

```
                                                        <BIT-POSITION>5</BIT-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_StorageStatus"/>
                                                  </PARAM>
                                                  <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                        <SHORT-
NAME>DTCRF</SHORT-NAME>
                                                        <LONG-NAME>DTC
Readyness Flag</LONG-NAME>
                                                        <BYTE-
POSITION>0</BYTE-POSITION>
                                                        <BIT-POSITION>4</BIT-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_ReadynessFlag"/>
                                                  </PARAM>
                                                  <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                        <SHORT-
NAME>DTCFS</SHORT-NAME>
                                                        <LONG-NAME>DTC Fault
Symptom</LONG-NAME>
                                                        <BYTE-
POSITION>0</BYTE-POSITION>
                                                        <BIT-POSITION>0</BIT-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_FaultSymptom"/>
                                                  </PARAM>
                                            </PARAMS>
                                      </STRUCTURE>
                                      <STRUCTURE ID="STRUCT_DTCAS">
                                            <SHORT-NAME>STRUCT_DTCAS</SHORT-
NAME>
                                            <LONG-NAME>DTC and Status</LONG-
NAME>
                                            <BYTE-SIZE>3</BYTE-SIZE>
                                            <PARAMS>
                                                  <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                  <SHORT-NAME>DTC</SHORT-NAME>
                                                  <BYTE-POSITION>0</BYTE-
POSITION>
                                                  <DOP-REF ID-
REF="DTCDOP_AllDTCs"/>
                                            </PARAM>
                                            <PARAM xsi:type="VALUE"
SEMANTIC="DATA">
                                                  <SHORT-
NAME>StatusOfDTC</SHORT-NAME>
                                                  <BYTE-POSITION>2</BYTE-
POSITION>
                                                  <DOP-REF ID-
REF="STRUCT_StatusOfDTC"/>
                                            </PARAM>
                                      </PARAMS>
                                </STRUCTURE>
                                <STRUCTURE ID="STRUCT_DDLIDItem" IS-
VISIBLE="false">
                                      <SHORT-NAME>STRUCT_DDLIDItem</SHORT-NAME>
                                      <PARAMS>
                                            <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                  <SHORT-
NAME>definitionMode</SHORT-NAME>
```

```
                                                    <BYTE-POSITION>0</BYTE-
POSITION>
                                                    <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
                                                </PARAM>
                                                <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                    <SHORT-
NAME>postionInDDLID</SHORT-NAME>
                                                    <BYTE-POSITION>1</BYTE-
POSITION>
                                                    <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
                                                </PARAM>
                                                <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                    <SHORT-
NAME>memorySize</SHORT-NAME>
                                                    <BYTE-POSITION>2</BYTE-
POSITION>
                                                    <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
                                                </PARAM>
                                                <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                    <SHORT-NAME>LID</SHORT-NAME>
                                                    <LONG-
NAME>localIdentifier</LONG-NAME>
                                                    <BYTE-POSITION>3</BYTE-
POSITION>
                                                    <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
                                                </PARAM>
                                                <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                    <SHORT-
NAME>postionInRecordLID</SHORT-NAME>
                                                    <BYTE-POSITION>4</BYTE-
POSITION>
                                                    <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
                                                </PARAM>
                                            </PARAMS>
                                        </STRUCTURE>
                                        <STRUCTURE ID="STRUCT_VehicleSpeedItem" IS-
VISIBLE="false">
                                            <SHORT-
NAME>STRUCT_VehicleSpeedItem</SHORT-NAME>
                                            <PARAMS>
                                                <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                    <SHORT-
NAME>definitionMode</SHORT-NAME>
                                                    <BYTE-POSITION>0</BYTE-
POSITION>
                                                    <CODED-VALUE>1</CODED-VALUE>
                                                    <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>8</BIT-
LENGTH>
                                                    </DIAG-CODED-TYPE>
                                                </PARAM>
                                                <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                    <SHORT-
NAME>postionInDDLID</SHORT-NAME>
```

```
                                                        <BYTE-POSITION>1</BYTE-
POSITION>
                                                        <CODED-VALUE>1</CODED-VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                                  <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                        <SHORT-
NAME>memorySize</SHORT-NAME>
                                                        <BYTE-POSITION>2</BYTE-
POSITION>
                                                        <CODED-VALUE>1</CODED-VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                                  <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                        <SHORT-NAME>LID</SHORT-NAME>
                                                        <LONG-
NAME>localIdentifier</LONG-NAME>
                                                        <BYTE-POSITION>3</BYTE-
POSITION>
                                                        <CODED-VALUE>2</CODED-VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                                  <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                        <SHORT-
NAME>postionInRecordLID</SHORT-NAME>
                                                        <BYTE-POSITION>4</BYTE-
POSITION>
                                                        <CODED-VALUE>1</CODED-VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                              <BIT-LENGTH>8</BIT-
LENGTH>
                                                        </DIAG-CODED-TYPE>
                                                  </PARAM>
                                            </PARAMS>
                                      </STRUCTURE>
                                      <STRUCTURE ID="STRUCT_ThrottlePositionItem" IS-
VISIBLE="false">
                                            <SHORT-
NAME>STRUCT_ThrottlePositionItem</SHORT-NAME>
                                            <LONG-NAME>Throttle Position Item</LONG-
NAME>
                                            <PARAMS>
                                                  <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                        <SHORT-
NAME>definitionMode</SHORT-NAME>
                                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                                        <CODED-VALUE>1</CODED-VALUE>
                                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
```

```
                                                        <BIT-LENGTH>8</BIT-
LENGTH>
                                                  </DIAG-CODED-TYPE>
                                            </PARAM>
                                            <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                  <SHORT-
NAME>postionInDDLID</SHORT-NAME>
                                                  <BYTE-POSITION>1</BYTE-
POSITION>
                                                  <CODED-VALUE>2</CODED-VALUE>
                                                  <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>8</BIT-
LENGTH>
                                                  </DIAG-CODED-TYPE>
                                            </PARAM>
                                            <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                  <SHORT-
NAME>memorySize</SHORT-NAME>
                                                  <BYTE-POSITION>2</BYTE-
POSITION>
                                                  <CODED-VALUE>1</CODED-VALUE>
                                                  <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>8</BIT-
LENGTH>
                                                  </DIAG-CODED-TYPE>
                                            </PARAM>
                                            <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                  <SHORT-NAME>LID</SHORT-NAME>
                                                  <LONG-
NAME>localIdentifier</LONG-NAME>
                                                  <BYTE-POSITION>3</BYTE-
POSITION>
                                                  <CODED-VALUE>1</CODED-VALUE>
                                                  <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>8</BIT-
LENGTH>
                                                  </DIAG-CODED-TYPE>
                                            </PARAM>
                                            <PARAM SEMANTIC="DATA"
xsi:type="CODED-CONST">
                                                  <SHORT-
NAME>postionInRecordLID</SHORT-NAME>
                                                  <BYTE-POSITION>4</BYTE-
POSITION>
                                                  <CODED-VALUE>5</CODED-VALUE>
                                                  <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                        <BIT-LENGTH>8</BIT-
LENGTH>
                                                  </DIAG-CODED-TYPE>
                                            </PARAM>
                                      </PARAMS>
                                </STRUCTURE>
                                <STRUCTURE ID="STRUCT_LID01" IS-VISIBLE="false">
                                      <SHORT-NAME>STRUCT_LID01</SHORT-NAME>
                                      <LONG-NAME>Local ID 01</LONG-NAME>
                                      <PARAMS>
                                            <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                  <SHORT-
NAME>EngineCoolantTemperature</SHORT-NAME>
```

```
                                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_EngineCoolantTemperature"/>
                                                  </PARAM>
                                                  <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                        <SHORT-
NAME>EngineSpeed</SHORT-NAME>
                                                        <BYTE-POSITION>1</BYTE-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_EngineSpeed"/>
                                                  </PARAM>
                                                  <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                        <SHORT-
NAME>ThrottlePosition</SHORT-NAME>
                                                        <BYTE-POSITION>3</BYTE-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_ThrottlePosition"/>
                                                  </PARAM>
                                            </PARAMS>
                                      </STRUCTURE>
                                      <STRUCTURE ID="STRUCT_LID02" IS-VISIBLE="false">
                                            <SHORT-NAME>STRUCT_LID02</SHORT-NAME>
                                            <LONG-NAME>Local ID 02</LONG-NAME>
                                            <PARAMS>
                                                  <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                        <SHORT-
NAME>VehicleSpeed</SHORT-NAME>
                                                        <BYTE-POSITION>0</BYTE-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_VehicleSpeed"/>
                                                  </PARAM>
                                                  <PARAM SEMANTIC="DATA"
xsi:type="VALUE">
                                                        <SHORT-
NAME>EngineSpeed</SHORT-NAME>
                                                        <BYTE-POSITION>2</BYTE-
POSITION>
                                                        <DOP-REF ID-
REF="DOP_EngineSpeed"/>
                                                  </PARAM>
                                            </PARAMS>
                                      </STRUCTURE>
                                </STRUCTURES>
                                <END-OF-PDU-FIELDS>
                                      <END-OF-PDU-FIELD ID="EOPDUF_ListOfDTCAS">
                                            <SHORT-NAME>EOPDUF_ListOfDTCAS</SHORT-
NAME>
                                            <LONG-NAME>List of DTC and Status</LONG-
NAME>
                                            <BASIC-STRUCTURE-REF ID-
REF="STRUCT_DTCAS"/>
                                      </END-OF-PDU-FIELD>
                                      <END-OF-PDU-FIELD ID="EOPDUF_EOPDU10">
                                            <SHORT-NAME>EOPDUF_EOPDU10</SHORT-NAME>
                                            <BASIC-STRUCTURE-REF ID-
REF="STRUCT_DDLIDItem"/>
                                            <MAX-NUMBER-OF-ITEMS>10</MAX-NUMBER-OF-
ITEMS>
                                      </END-OF-PDU-FIELD>
                                </END-OF-PDU-FIELDS>
```

```
                        <UNIT-SPEC>
                              <UNITS>
                                    <UNIT ID="DegreeCelsius">
                                          <SHORT-NAME>DegreeCelsius</SHORT-
NAME>
                                          <LONG-NAME>Degree Celsius</LONG-
NAME>
                                          <DISPLAY-
NAME>&#194;&#176;C</DISPLAY-NAME>
                                          <FACTOR-SI-TO-UNIT>1</FACTOR-SI-TO-
UNIT>
                                          <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-
UNIT>
                                          <PHYSICAL-DIMENSION-REF ID-
REF="PD_DegreeCelsius"/>
                                    </UNIT>
                                    <UNIT ID="Kmh">
                                          <SHORT-NAME>Kmh</SHORT-NAME>
                                          <LONG-NAME>Kilometers per
Hour</LONG-NAME>
                                          <DISPLAY-NAME>km/h</DISPLAY-NAME>
                                          <FACTOR-SI-TO-UNIT>3.6</FACTOR-SI-
TO-UNIT>
                                          <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-
UNIT>
                                          <PHYSICAL-DIMENSION-REF ID-
REF="PD_Velocity"/>
                                    </UNIT>
                                    <UNIT ID="Rpm">
                                          <SHORT-NAME>Rpm</SHORT-NAME>
                                          <LONG-NAME>Rounds Per Minute</LONG-
NAME>
                                          <DISPLAY-NAME>1/s</DISPLAY-NAME>
                                          <FACTOR-SI-TO-UNIT>60</FACTOR-SI-
TO-UNIT>
                                          <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-
UNIT>
                                          <PHYSICAL-DIMENSION-REF ID-
REF="PD_Frequency"/>
                                    </UNIT>
                                    <UNIT ID="Km">
                                          <SHORT-NAME>Km</SHORT-NAME>
                                          <LONG-NAME>Kilometers</LONG-NAME>
                                          <DISPLAY-NAME>km</DISPLAY-NAME>
                                          <FACTOR-SI-TO-UNIT>0.001</FACTOR-
SI-TO-UNIT>
                                          <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-
UNIT>
                                          <PHYSICAL-DIMENSION-REF ID-
REF="PD_Meter"/>
                                    </UNIT>
                                    <UNIT ID="Volt">
                                          <SHORT-NAME>Volt</SHORT-NAME>
                                          <LONG-NAME>Volt</LONG-NAME>
                                          <DISPLAY-NAME>V</DISPLAY-NAME>
                                          <FACTOR-SI-TO-UNIT>1</FACTOR-SI-TO-
UNIT>
                                          <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-
UNIT>
                                          <PHYSICAL-DIMENSION-REF ID-
REF="PD_Volt"/>
                                    </UNIT>
                              </UNITS>
                              <PHYSICAL-DIMENSIONS>
                                    <PHYSICAL-DIMENSION
ID="PD_DegreeCelsius">
```

```
                                                <SHORT-
NAME>PD_DegreeCelsius</SHORT-NAME>
                                                <LONG-NAME>Degree Celsius</LONG-
NAME>
                                                <TEMPERATURE-EXP>1</TEMPERATURE-
EXP>
                                          </PHYSICAL-DIMENSION>
                                          <PHYSICAL-DIMENSION ID="PD_Velocity">
                                                <SHORT-NAME>PD_Velocity</SHORT-
NAME>
                                                <LONG-NAME>Meter Per Second</LONG-
NAME>
                                                <LENGTH-EXP>1</LENGTH-EXP>
                                                <TIME-EXP>-1</TIME-EXP>
                                          </PHYSICAL-DIMENSION>
                                          <PHYSICAL-DIMENSION ID="PD_Frequency">
                                                <SHORT-NAME>PD_Frequency</SHORT-
NAME>
                                                <LONG-NAME>Per Second</LONG-NAME>
                                                <TIME-EXP>-1</TIME-EXP>
                                          </PHYSICAL-DIMENSION>
                                          <PHYSICAL-DIMENSION ID="PD_Meter">
                                                <SHORT-NAME>PD_Meter</SHORT-NAME>
                                                <LONG-NAME>Meter</LONG-NAME>
                                                <LENGTH-EXP>1</LENGTH-EXP>
                                          </PHYSICAL-DIMENSION>
                                          <PHYSICAL-DIMENSION ID="PD_Volt">
                                                <SHORT-NAME>PD_Volt</SHORT-NAME>
                                                <LONG-NAME>Volt</LONG-NAME>
                                                <LENGTH-EXP>2</LENGTH-EXP>
                                                <MASS-EXP>1</MASS-EXP>
                                                <TIME-EXP>-3</TIME-EXP>
                                          </PHYSICAL-DIMENSION>
                                    </PHYSICAL-DIMENSIONS>
                              </UNIT-SPEC>
                              <TABLES>
                                    <TABLE ID="TAB_LIDs" SEMANTIC="LOCAL-ID">
                                          <SHORT-NAME>TAB_LIDs</SHORT-NAME>
                                          <LONG-NAME>Table to interprete Local ID
and according measurements</LONG-NAME>
                                          <TABLE-ROW ID="TABROW_LID01">
                                                <SHORT-NAME>TABROW_LID01</SHORT-
NAME>
                                                <LONG-NAME>Local ID 01</LONG-NAME>
                                                <KEY>1</KEY>
                                                <STRUCTURE-REF ID-
REF="STRUCT_LID01"/>
                                          </TABLE-ROW>
                                          <TABLE-ROW ID="TABROW_LID02">
                                                <SHORT-NAME>TABROW_LID02</SHORT-
NAME>
                                                <LONG-NAME>Local ID 02</LONG-NAME>
                                                <KEY>2</KEY>
                                                <STRUCTURE-REF ID-
REF="STRUCT_LID02"/>
                                          </TABLE-ROW>
                                    </TABLE>
                              </TABLES>
                        </DIAG-DATA-DICTIONARY-SPEC>
                        <DIAG-COMMS>
<DIAG-SERVICE ID="DS_ReadStatusOfDTC" SEMANTIC="FAULTREAD" ADDRESSING="PHYSICAL">
      <SHORT-NAME>DS_ReadStatusOfDTC</SHORT-NAME>
      <LONG-NAME>Read Status of DTC</LONG-NAME>
      <AUDIENCE IS-DEVELOPMENT="true" IS-AFTERMARKET="true" IS-
MANUFACTURING="true"/>
      <REQUEST-REF ID-REF="REQ_RSODTC"/>
      <POS-RESPONSE-REFS>
```

```
            <POS-RESPONSE-REF ID-REF="RESP_RSODTC"/>
      </POS-RESPONSE-REFS>
</DIAG-SERVICE>
                        <DIAG-SERVICE ID="DS_ReadDTCByStatus"
SEMANTIC="FAULTREAD" IS-MANDATORY="true" ADDRESSING="PHYSICAL">
                              <SHORT-NAME>DS_ReadDTCByStatus</SHORT-NAME>
                              <LONG-NAME>Read DTC By Status</LONG-NAME>
                              <AUDIENCE IS-DEVELOPMENT="true" IS-
AFTERMARKET="true" IS-MANUFACTURING="true"/>
                              <REQUEST-REF ID-REF="REQ_RDTCBS"/>
                              <POS-RESPONSE-REFS>
                                    <POS-RESPONSE-REF ID-REF="RESP_RDTCBS"/>
                              </POS-RESPONSE-REFS>
                        </DIAG-SERVICE>
                        <DIAG-SERVICE ID="DS_DDLID_READ" SEMANTIC="STOREDDATA"
DIAGNOSTIC-CLASS="READ-DYN-DEFINED-MESSAGE" ADDRESSING="PHYSICAL">
                              <SHORT-NAME>DS_DDLID_READ</SHORT-NAME>
                              <LONG-NAME>Read Data By Identifer - Dynamically
Defined Local ID</LONG-NAME>
                              <AUDIENCE IS-AFTERMARKET="true" IS-
DEVELOPMENT="true" IS-MANUFACTURING="true" IS-SUPPLIER="true" IS-
AFTERSALES="true"/>
                              <REQUEST-REF ID-REF="REQ_DDLID_READ"/>
                              <POS-RESPONSE-REFS>
                                    <POS-RESPONSE-REF ID-
REF="RESP_DDLID_READ"/>
                              </POS-RESPONSE-REFS>
                        </DIAG-SERVICE>
                        <DIAG-SERVICE ID="DS_DDLID_STATIC"
SEMANTIC="STOREDDATA" DIAGNOSTIC-CLASS="DYN-DEF-MESSAGE" ADDRESSING="PHYSICAL">
                              <SHORT-NAME>DS_DDLID_STATIC</SHORT-NAME>
                              <AUDIENCE IS-AFTERMARKET="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-MANUFACTURING="true" IS-
SUPPLIER="true"/>
                              <REQUEST-REF ID-REF="REQ_DDLID_STATIC"/>
                              <POS-RESPONSE-REFS>
                                    <POS-RESPONSE-REF ID-REF="RESP_DDLID"/>
                              </POS-RESPONSE-REFS>
                        </DIAG-SERVICE>
                        <DIAG-SERVICE ID="DS_DDLID_DYNAMIC"
SEMANTIC="STOREDDATA" DIAGNOSTIC-CLASS="DYN-DEF-MESSAGE" ADDRESSING="PHYSICAL">
                              <SHORT-NAME>DS_DDLID_DYNAMIC</SHORT-NAME>
                              <AUDIENCE IS-AFTERMARKET="true" IS-
AFTERSALES="true" IS-DEVELOPMENT="true" IS-MANUFACTURING="true" IS-
SUPPLIER="true"/>
                              <REQUEST-REF ID-REF="REQ_DDLID_DYNAMIC"/>
                              <POS-RESPONSE-REFS>
                                    <POS-RESPONSE-REF ID-REF="RESP_DDLID"/>
                              </POS-RESPONSE-REFS>
                        </DIAG-SERVICE>
                        <DIAG-SERVICE ID="DS_DDLID_CLEAR" DIAGNOSTIC-
CLASS="CLEAR-DYN-DEF-MESSAGE">
                              <SHORT-NAME>DS_DDLID_CLEAR</SHORT-NAME>
                              <AUDIENCE/>
                              <REQUEST-REF ID-REF="REQ_DDLID_CLEAR"/>
                              <POS-RESPONSE-REFS>
                                    <POS-RESPONSE-REF ID-
REF="RESP_DDLID_CLEAR"/>
                              </POS-RESPONSE-REFS>
                        </DIAG-SERVICE>
                  </DIAG-COMMS>
                  <REQUESTS>
                        <!--+++++++++++++++ DTC requests: +++++++++++++++-->
                        <REQUEST ID="REQ_RSODTC">
                              <SHORT-NAME>REQ_RSODTC</SHORT-NAME>
                              <LONG-NAME>Read Status of DTC Request</LONG-
NAME>
```

```
                                    <PARAMS>
                                          <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                                <SHORT-NAME>ServiceID</SHORT-NAME>
                                                <LONG-NAME>Read Status of DTC
Request</LONG-NAME>
                                                <BYTE-POSITION>0</BYTE-POSITION>
                                                <CODED-VALUE>23</CODED-VALUE>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                      <BIT-LENGTH>8</BIT-LENGTH>
                                                </DIAG-CODED-TYPE>
                                          </PARAM>
                                          <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                                                <SHORT-NAME>GroupOfDTC</SHORT-NAME>
                                                <LONG-NAME>Group of DTC</LONG-NAME>
                                                <BYTE-POSITION>1</BYTE-POSITION>
                                                <DOP-REF ID-REF="DOP_GroupOfDTC"/>
                                          </PARAM>
                                    </PARAMS>
                              </REQUEST>
                              <REQUEST ID="REQ_RDTCBS">
                                    <SHORT-NAME>REQ_RDTCBS</SHORT-NAME>
                                    <LONG-NAME>Read DTC By Status Request</LONG-
NAME>
                                    <PARAMS>
                                          <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                                <SHORT-NAME>ServiceID</SHORT-NAME>
                                                <LONG-NAME>Read DTC By Status
</LONG-NAME>
                                                <BYTE-POSITION>0</BYTE-POSITION>
                                                <CODED-VALUE>24</CODED-VALUE>
                                                <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                      <BIT-LENGTH>8</BIT-LENGTH>
                                                </DIAG-CODED-TYPE>
                                          </PARAM>
                                          <PARAM xsi:type="VALUE"
SEMANTIC="OPTION">
                                                <SHORT-NAME>StatusOfDTCReq</SHORT-
NAME>
                                                <LONG-NAME>Status of DTC in
request</LONG-NAME>
                                                <BYTE-POSITION>1</BYTE-POSITION>
                                                <PHYSICAL-DEFAULT-VALUE>request
identified 2 Byte Hex DTC and Status</PHYSICAL-DEFAULT-VALUE>
                                                <DOP-REF ID-
REF="DOP_StatusOfDTCReq"/>
                                          </PARAM>
                                          <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                                                <SHORT-NAME>GroupOfDTC</SHORT-NAME>
                                                <LONG-NAME>Group of DTC</LONG-NAME>
                                                <BYTE-POSITION>2</BYTE-POSITION>
                                                <PHYSICAL-DEFAULT-VALUE>All
DTCs</PHYSICAL-DEFAULT-VALUE>
                                                <DOP-REF ID-REF="DOP_GroupOfDTC"/>
                                          </PARAM>
                                    </PARAMS>
                              </REQUEST>
                              <!--+++++++++++++++ DDLID requests: +++++++++++++++-->
                              <REQUEST ID="REQ_DDLID_STATIC">
                                    <SHORT-NAME>REQ_DDLID_STATIC</SHORT-NAME>
                                    <LONG-NAME>Dynamically Defined Local ID Request
- static version</LONG-NAME>
                                    <PARAMS>
```

```
                                        <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                              <SHORT-NAME>ServiceID</SHORT-NAME>
                                              <BYTE-POSITION>0</BYTE-POSITION>
                                              <CODED-VALUE>44</CODED-VALUE>
                                              <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>8</BIT-LENGTH>
                                              </DIAG-CODED-TYPE>
                                        </PARAM>
                                        <PARAM xsi:type="VALUE" SEMANTIC="DYN-
ID">
                                              <SHORT-NAME>DDLID</SHORT-NAME>
                                              <LONG-NAME>Dynamically Defined
Local ID</LONG-NAME>
                                              <BYTE-POSITION>1</BYTE-POSITION>
                                              <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
                                        </PARAM>
                                        <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                                              <SHORT-NAME>Item1</SHORT-NAME>
                                              <BYTE-POSITION>2</BYTE-POSITION>
                                              <DOP-REF ID-
REF="STRUCT_VehicleSpeedItem"/>
                                        </PARAM>
                                        <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                                              <SHORT-NAME>Item2</SHORT-NAME>
                                              <BYTE-POSITION>7</BYTE-POSITION>
                                              <DOP-REF ID-
REF="STRUCT_ThrottlePositionItem"/>
                                        </PARAM>
                                  </PARAMS>
                            </REQUEST>
                            <REQUEST ID="REQ_DDLID_DYNAMIC">
                                  <SHORT-NAME>REQ_DDLID_DYNAMIC</SHORT-NAME>
                                  <LONG-NAME>Dynamically Defined Local ID Request
- dynamic version</LONG-NAME>
                                  <PARAMS>
                                        <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                              <SHORT-NAME>ServiceID</SHORT-NAME>
                                              <LONG-NAME>Dynamically Define Data
Identifier</LONG-NAME>
                                              <BYTE-POSITION>0</BYTE-POSITION>
                                              <CODED-VALUE>44</CODED-VALUE>
                                              <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>8</BIT-LENGTH>
                                              </DIAG-CODED-TYPE>
                                        </PARAM>
                                        <PARAM xsi:type="VALUE" SEMANTIC="DYN-
ID">
                                              <SHORT-NAME>DDLID</SHORT-NAME>
                                              <LONG-NAME>Dynamically Defined
Local ID</LONG-NAME>
                                              <BYTE-POSITION>1</BYTE-POSITION>
                                              <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
                                        </PARAM>
                                        <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                                              <SHORT-NAME>DDLIDContent</SHORT-
NAME>
                                              <LONG-NAME>DDLIDContent</LONG-NAME>
                                              <BYTE-POSITION>2</BYTE-POSITION>
                                              <DOP-REF ID-REF="EOPDUF_EOPDU10"/>
                                        </PARAM>
                                  </PARAMS>
```

```
                        </REQUEST>
                        <REQUEST ID="REQ_DDLID_READ">
                              <SHORT-NAME>REQ_DDLID_READ</SHORT-NAME>
                              <LONG-NAME>Read Data By ID - Dynmically Defined
Local ID Request</LONG-NAME>
                              <PARAMS>
                                    <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                          <SHORT-NAME>ServiceID</SHORT-NAME>
                                          <LONG-NAME>Read Data By ID
Request</LONG-NAME>
                                          <BYTE-POSITION>0</BYTE-POSITION>
                                          <CODED-VALUE>34</CODED-VALUE>
                                          <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                <BIT-LENGTH>8</BIT-LENGTH>
                                          </DIAG-CODED-TYPE>
                                    </PARAM>
                                    <PARAM xsi:type="VALUE" SEMANTIC="DYN-
ID">
                                          <SHORT-NAME>DDLID</SHORT-NAME>
                                          <LONG-NAME>Dynamically Defined
Local ID</LONG-NAME>
                                          <BYTE-POSITION>1</BYTE-POSITION>
                                          <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
                                    </PARAM>
                              </PARAMS>
                        </REQUEST>
                        <REQUEST ID="REQ_DDLID_CLEAR">
                              <SHORT-NAME>REQ_DDLID_CLEAR</SHORT-NAME>
                              <PARAMS>
                                    <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                          <SHORT-NAME>ServiceID</SHORT-NAME>
                                          <LONG-NAME>Read Data By ID
Request</LONG-NAME>
                                          <BYTE-POSITION>0</BYTE-POSITION>
                                          <CODED-VALUE>34</CODED-VALUE>
                                          <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                <BIT-LENGTH>8</BIT-LENGTH>
                                          </DIAG-CODED-TYPE>
                                    </PARAM>
                                    <PARAM xsi:type="VALUE" SEMANTIC="DYN-
ID">
                                          <SHORT-NAME>DDLID</SHORT-NAME>
                                          <LONG-NAME>Dynamically Defined
Local ID</LONG-NAME>
                                          <BYTE-POSITION>1</BYTE-POSITION>
                                          <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
                                    </PARAM>
                                    <PARAM xsi:type="CODED-CONST">
                                          <SHORT-NAME>definitionMode</SHORT-
NAME>
                                          <BYTE-POSITION>2</BYTE-POSITION>
                                          <CODED-VALUE>4</CODED-VALUE>
                                          <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                <BIT-LENGTH>8</BIT-LENGTH>
                                          </DIAG-CODED-TYPE>
                                    </PARAM>
                              </PARAMS>
                        </REQUEST>
                  </REQUESTS>
                  <POS-RESPONSES>
```

```
                        <!--+++++++++++++++ DTC responses: +++++++++++++++-->
<POS-RESPONSE ID="RESP_RSODTC">
      <SHORT-NAME>RESP_RSODTC</SHORT-NAME>
      <LONG-NAME>Read Status of DTC Response</LONG-NAME>
      <PARAMS>
            <PARAM xsi:type="CODED-CONST" SEMANTIC="SERVICE-ID">
                  <SHORT-NAME>ServiceID</SHORT-NAME>
                  <LONG-NAME>Read Status of DTC Response</LONG-NAME>
                  <BYTE-POSITION>0</BYTE-POSITION>
                  <CODED-VALUE>87</CODED-VALUE>
                  <DIAG-CODED-TYPE BASE-DATA-TYPE="A_UINT32"
xsi:type="STANDARD-LENGTH-TYPE">
                        <BIT-LENGTH>8</BIT-LENGTH>
                  </DIAG-CODED-TYPE>
            </PARAM>
            <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                  <SHORT-NAME>NumberOfDTC</SHORT-NAME>
                  <LONG-NAME>number of DTC</LONG-NAME>
                  <BYTE-POSITION>1</BYTE-POSITION>
                  <DOP-REF ID-REF="DOP_1ByteIdentical"/>
            </PARAM>
            <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                  <SHORT-NAME>SwitchKeyDTC</SHORT-NAME>
                  <LONG-NAME>DTC SwitchKEY</LONG-NAME>
                  <BYTE-POSITION>2</BYTE-POSITION>
                  <DOP-REF ID-REF="DOP_2ByteIdentical"/>
            </PARAM>
            <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                  <SHORT-NAME>DTCAS</SHORT-NAME>
                  <LONG-NAME>DTC and Status</LONG-NAME>
                  <BYTE-POSITION>2</BYTE-POSITION>
                  <DOP-REF ID-REF="STRUCT_DTCAS"/>
            </PARAM>
            <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                  <SHORT-NAME>ENVDATA</SHORT-NAME>
                  <LONG-NAME>Environmental Data</LONG-NAME>
                  <BYTE-POSITION>5</BYTE-POSITION>
                  <DOP-REF ID-REF="ENVDESC_EnvDataDesc"/>
            </PARAM>
      </PARAMS>
</POS-RESPONSE>
                        <POS-RESPONSE ID="RESP_RDTCBS">
                              <SHORT-NAME>RESP_RDTCBS</SHORT-NAME>
                              <LONG-NAME>Read DTC By Status Response</LONG-
NAME>
                              <PARAMS>
                                    <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                          <SHORT-NAME>ServiceID</SHORT-NAME>
                                          <LONG-NAME>Read DTC By Status
Response</LONG-NAME>
                                          <BYTE-POSITION>0</BYTE-POSITION>
                                          <CODED-VALUE>88</CODED-VALUE>
                                          <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                <BIT-LENGTH>8</BIT-LENGTH>
                                          </DIAG-CODED-TYPE>
                                    </PARAM>
                                    <PARAM xsi:type="VALUE"
SEMANTIC="INFOTYPE">
                                          <SHORT-NAME>NumberOfDTC</SHORT-
NAME>
                                          <LONG-NAME>Number Of DTC</LONG-
NAME>
                                          <BYTE-POSITION>1</BYTE-POSITION>
                                          <DOP-REF ID-
REF="DOP_1ByteIdentical"/>
```

```
                                        </PARAM>
                                        <PARAM xsi:type="VALUE" SEMANTIC="DATA">
                                              <SHORT-NAME>ListOfDTCAS</SHORT-
NAME>
                                              <LONG-NAME>List of DTC and
Status</LONG-NAME>
                                              <BYTE-POSITION>2</BYTE-POSITION>
                                              <DOP-REF ID-
REF="EOPDUF_ListOfDTCAS"/>
                                        </PARAM>
                                  </PARAMS>
                            </POS-RESPONSE>
                            <!--+++++++++++++++ DDLID responses: +++++++++++++++-->
                            <POS-RESPONSE ID="RESP_DDLID">
                                  <SHORT-NAME>RESP_DDLID</SHORT-NAME>
                                  <LONG-NAME>Dynamically Defined Local ID
Response</LONG-NAME>
                                  <PARAMS>
                                        <PARAM SEMANTIC="SERVICE-ID"
xsi:type="CODED-CONST">
                                              <SHORT-NAME>ServiceID</SHORT-NAME>
                                              <LONG-NAME>Dynamically Define Data
ID</LONG-NAME>
                                              <BYTE-POSITION>0</BYTE-POSITION>
                                              <CODED-VALUE>108</CODED-VALUE>
                                              <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>8</BIT-LENGTH>
                                              </DIAG-CODED-TYPE>
                                        </PARAM>
                                        <PARAM SEMANTIC="DYN-ID"
xsi:type="MATCHING-REQUEST-PARAM">
                                              <SHORT-NAME>DDDID</SHORT-NAME>
                                              <LONG-NAME>Dynymically Defined
Local ID</LONG-NAME>
                                              <BYTE-POSITION>1</BYTE-POSITION>
                                              <REQUEST-BYTE-POS>1</REQUEST-BYTE-
POS>
                                              <BYTE-LENGTH>1</BYTE-LENGTH>
                                        </PARAM>
                                  </PARAMS>
                            </POS-RESPONSE>
                            <POS-RESPONSE ID="RESP_DDLID_READ">
                                  <SHORT-NAME>RESP_DDLID_READ</SHORT-NAME>
                                  <LONG-NAME>Read Data By Local ID Response</LONG-
NAME>
                                  <PARAMS>
                                        <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                              <SHORT-NAME>ServiceID</SHORT-NAME>
                                              <LONG-NAME>Read Data By Local ID
Response</LONG-NAME>
                                              <BYTE-POSITION>0</BYTE-POSITION>
                                              <CODED-VALUE>98</CODED-VALUE>
                                              <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                                    <BIT-LENGTH>8</BIT-LENGTH>
                                              </DIAG-CODED-TYPE>
                                        </PARAM>
                                        <PARAM SEMANTIC="DATA"
xsi:type="MATCHING-REQUEST-PARAM">
                                              <SHORT-NAME>DDLID</SHORT-NAME>
                                              <LONG-NAME>Dynamically Defined
Local ID</LONG-NAME>
                                              <BYTE-POSITION>1</BYTE-POSITION>
                                              <REQUEST-BYTE-POS>1</REQUEST-BYTE-
POS>
```

```
                                        <BYTE-LENGTH>1</BYTE-LENGTH>
                                    </PARAM>
                                    <PARAM SEMANTIC="DATA"
xsi:type="DYNAMIC">
                                        <SHORT-NAME>DDLIDContent</SHORT-
NAME>
                                        <BYTE-POSITION>2</BYTE-POSITION>
                                    </PARAM>
                                </PARAMS>
                            </POS-RESPONSE>
                            <POS-RESPONSE ID="RESP_DDLID_CLEAR">
                                <SHORT-NAME>RESP_DDLID_CLEAR</SHORT-NAME>
                                <PARAMS>
                                    <PARAM xsi:type="CODED-CONST"
SEMANTIC="SERVICE-ID">
                                        <SHORT-NAME>ServiceID</SHORT-NAME>
                                        <LONG-NAME>Read Data By ID
Request</LONG-NAME>
                                        <BYTE-POSITION>0</BYTE-POSITION>
                                        <CODED-VALUE>98</CODED-VALUE>
                                        <DIAG-CODED-TYPE BASE-DATA-
TYPE="A_UINT32" xsi:type="STANDARD-LENGTH-TYPE">
                                            <BIT-LENGTH>8</BIT-LENGTH>
                                        </DIAG-CODED-TYPE>
                                    </PARAM>
                                    <PARAM xsi:type="MATCHING-REQUEST-PARAM"
SEMANTIC="DATA">
                                        <SHORT-NAME>DDLID</SHORT-NAME>
                                        <LONG-NAME>Dynamically Defined
Local ID</LONG-NAME>
                                        <BYTE-POSITION>1</BYTE-POSITION>
                                        <REQUEST-BYTE-POS>1</REQUEST-BYTE-
POS>
                                        <BYTE-LENGTH>1</BYTE-LENGTH>
                                    </PARAM>
                                </PARAMS>
                            </POS-RESPONSE>
                        </POS-RESPONSES>
                        <COMPARAM-SPEC-REF ID-REF="COMPARAM-SPEC.ISO14230_CPS"
DOCREF="ISO14230_CPS" DOCTYPE="COMPARAM-SPEC"/>
                    </PROTOCOL>
                </PROTOCOLS>
                <BASE-VARIANTS>
                    <BASE-VARIANT ID="BV">
                        <SHORT-NAME>BV</SHORT-NAME>
                        <DYN-DEFINED-SPEC>
                            <DYN-ID-DEF-MODE-INFOS>
                                <DYN-ID-DEF-MODE-INFO>
                                    <DEF-MODE>LOCAL-ID</DEF-MODE>
                                    <CLEAR-DYN-DEF-MESSAGE-REF ID-
REF="DS_DDLID_CLEAR"/>
                                    <READ-DYN-DEF-MESSAGE-REF ID-
REF="DS_DDLID_READ"/>
                                    <DYN-DEF-MESSAGE-REF ID-
REF="DS_DDLID_DYNAMIC"/>
                                    <SUPPORTED-DYN-IDS>
                                        <SUPPORTED-DYN-ID>F0</SUPPORTED-
DYN-ID>
                                        <SUPPORTED-DYN-ID>F1</SUPPORTED-
DYN-ID>
                                    </SUPPORTED-DYN-IDS>
                                    <SELECTION-TABLE-REFS>
                                        <SELECTION-TABLE-REF ID-
REF="TAB_LIDs"/>
                                    </SELECTION-TABLE-REFS>
                                </DYN-ID-DEF-MODE-INFO>
                            </DYN-ID-DEF-MODE-INFOS>
```

```
                    </DYN-DEFINED-SPEC>
                    <PARENT-REFS>
                          <PARENT-REF ID-REF="ISO14230"/>
                    </PARENT-REFS>
              </BASE-VARIANT>
       </BASE-VARIANTS>
</DIAG-LAYER-CONTAINER>
</ODX>
```

C.3 ECU-MEM

In the following, one ECU-MEM structure is implemented with an ECU-MEM-CONNECTOR with descriptions of SESSIONs defined in this ECU-MEM.

```
<?xml version="1.0" encoding="UTF-8"?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" MODEL-VERSION=„2.1.0“>
       <FLASH ID="FLASH_ECU_ID101">
              <SHORT-NAME>FLASH_ECU</SHORT-NAME>
              <ECU-MEMS>
              <ECU-MEM ID="FLASH_ECU_ID102">
                    <SHORT-NAME>ECUMEM_FLASH_ECU</SHORT-NAME>
                    <MEM>
                          <SESSIONS>
                                <SESSION ID="FLASH_ECU_ID110">
                                      <SHORT-
NAME>FLASH_ECU_SW_V01_ON_HW_V01</SHORT-NAME>
                                      <LONG-NAME>Progamm Code and Data Segments
with software V01 on hardware V01</LONG-NAME>
                                      <EXPECTED-IDENTS>
                                            <EXPECTED-IDENT
ID="FLASH_ECU_ID112">
                                                  <SHORT-
NAME>HardwareVersion</SHORT-NAME>
                                                  <IDENT-VALUES>
                                                        <IDENT-VALUE
TYPE="A_BYTEFIELD">V01</IDENT-VALUE>
                                                  </IDENT-VALUES>
                                            </EXPECTED-IDENT>
                                      </EXPECTED-IDENTS>
                                      <SECURITYS>
                                            <SECURITY>
                                                  <SECURITY-METHOD
TYPE="A_ASCIISTRING">SECMETH_SIG01</SECURITY-METHOD>
                                                  <FW-SIGNATURE
TYPE="A_BYTEFIELD">0A5F37</FW-SIGNATURE>
                                            </SECURITY>
                                      </SECURITYS>
                                      <DATABLOCK-REFS>
                                            <DATABLOCK-REF ID-
REF="FLASH_ECU_ID150"/>
                                            <DATABLOCK-REF ID-
REF="FLASH_ECU_ID151"/>
                                            <DATABLOCK-REF ID-
REF="FLASH_ECU_ID153"/>
                                            <DATABLOCK-REF ID-
REF="FLASH_ECU_ID154"/>
                                      </DATABLOCK-REFS>
                                </SESSION>
                                <SESSION ID="FLASH_ECU_ID115">
                                      <SHORT-NAME>UPDATE_DATA_V02</SHORT-NAME>
                                      <LONG-NAME>Update both Data Segments to
V02</LONG-NAME>
                                      <CHECKSUMS>
                                            <CHECKSUM ID="FLASH_ECU_ID117">
                                                  <SHORT-
NAME>CHECKSUM_CPU1_DB02</SHORT-NAME>
```

```
                                                        <FILLBYTE>FF</FILLBYTE>
                                                        <SOURCE-START-
ADDRESS>010000</SOURCE-START-ADDRESS>
                                                        <CHECKSUM-
ALG>CHECKALGO_01</CHECKSUM-ALG>
                                                        <SOURCE-END-
ADDRESS>0137FF</SOURCE-END-ADDRESS>
                                                        <CHECKSUM-RESULT
TYPE="A_BYTEFIELD">55B0</CHECKSUM-RESULT>
                                                  </CHECKSUM>
                                                  <CHECKSUM ID="FLASH_ECU_ID118">
                                                        <SHORT-
NAME>CHECKSUM_CPU2_DB02</SHORT-NAME>
                                                        <FILLBYTE>00</FILLBYTE>
                                                        <SOURCE-START-
ADDRESS>6000</SOURCE-START-ADDRESS>
                                                        <CHECKSUM-
ALG>CHECKALGO_02</CHECKSUM-ALG>
                                                        <SOURCE-END-
ADDRESS>7FFF</SOURCE-END-ADDRESS>
                                                        <CHECKSUM-RESULT
TYPE="A_BYTEFIELD">030E</CHECKSUM-RESULT>
                                                  </CHECKSUM>
                                            </CHECKSUMS>
                                            <DATABLOCK-REFS>
                                                  <DATABLOCK-REF ID-
REF="FLASH_ECU_ID163"/>
                                                  <DATABLOCK-REF ID-
REF="FLASH_ECU_ID160"/>
                                            </DATABLOCK-REFS>
                                      </SESSION>
                                      <SESSION ID="FLASH_ECU_ID120">
                                            <SHORT-
NAME>UPDATE_CPU1_HW_V02_WITH_SW_V02</SHORT-NAME>
                                            <LONG-NAME>Update CPU1 Code and Data
Segments to V02</LONG-NAME>
                                            <EXPECTED-IDENTS>
                                                  <EXPECTED-IDENT
ID="FLASH_ECU_ID122">
                                                        <SHORT-
NAME>HardwareVersion</SHORT-NAME>
                                                        <IDENT-VALUES>
                                                              <IDENT-VALUE
TYPE="A_BYTEFIELD">V02</IDENT-VALUE>
                                                        </IDENT-VALUES>
                                                  </EXPECTED-IDENT>
                                            </EXPECTED-IDENTS>
                                            <CHECKSUMS>
                                                  <CHECKSUM ID="FLASH_ECU_ID123">
                                                        <SHORT-
NAME>CHECKSUM_CPU1_DB02</SHORT-NAME>
                                                        <FILLBYTE>FF</FILLBYTE>
                                                        <SOURCE-START-
ADDRESS>010000</SOURCE-START-ADDRESS>
                                                        <CHECKSUM-
ALG>CHECKALGO_01</CHECKSUM-ALG>
                                                        <SOURCE-END-
ADDRESS>013FFF</SOURCE-END-ADDRESS>
                                                        <CHECKSUM-RESULT
TYPE="A_BYTEFIELD">55B0</CHECKSUM-RESULT>
                                                  </CHECKSUM>
                                            </CHECKSUMS>
                                            <DATABLOCK-REFS>
                                                  <DATABLOCK-REF ID-
REF="FLASH_ECU_ID162"/>
                                                  <DATABLOCK-REF ID-
REF="FLASH_ECU_ID163"/>
```

```
                                        </DATABLOCK-REFS>
                                    </SESSION>
                                    <SESSION ID="FLASH_ECU_ID124">
                                        <SHORT-
NAME>FLASH_ECU_SW_V02_ON_HW_V02</SHORT-NAME>
                                        <LONG-NAME>Progamm Code and Data Segments
with software V02 on hardware V02</LONG-NAME>
                                        <EXPECTED-IDENTS>
                                            <EXPECTED-IDENT
ID="FLASH_ECU_ID125">
                                                <SHORT-
NAME>HardwareVersion</SHORT-NAME>
                                                <IDENT-VALUES>
                                                    <IDENT-VALUE
TYPE="A_BYTEFIELD">V02</IDENT-VALUE>
                                                </IDENT-VALUES>
                                            </EXPECTED-IDENT>
                                        </EXPECTED-IDENTS>
                                        <SECURITYS>
                                            <SECURITY>
                                                <SECURITY-METHOD
TYPE="A_ASCIISTRING">SECMETH_SIG01</SECURITY-METHOD>
                                                <FW-SIGNATURE
TYPE="A_BYTEFIELD">0A5F37</FW-SIGNATURE>
                                            </SECURITY>
                                        </SECURITYS>
                                        <DATABLOCK-REFS>
                                            <DATABLOCK-REF ID-
REF="FLASH_ECU_ID156"/>
                                            <DATABLOCK-REF ID-
REF="FLASH_ECU_ID157"/>
                                            <DATABLOCK-REF ID-
REF="FLASH_ECU_ID159"/>
                                            <DATABLOCK-REF ID-
REF="FLASH_ECU_ID160"/>
                                        </DATABLOCK-REFS>
                                    </SESSION>
                                </SESSIONS>
                                <DATABLOCKS>
                                    <DATABLOCK ID="FLASH_ECU_ID150" TYPE="CODE">
                                        <SHORT-NAME>CODE_CPU1_V01</SHORT-NAME>
                                        <FLASHDATA-REF ID-REF="FLASH_ECU_ID170"/>
                                    </DATABLOCK>
                                    <DATABLOCK ID="FLASH_ECU_ID151" TYPE="DATA">
                                        <SHORT-NAME>DATA_CPU1_V01</SHORT-NAME>
                                        <FLASHDATA-REF ID-REF="FLASH_ECU_ID171"/>
                                        <SEGMENTS>
                                            <SEGMENT ID="FLASH_ECU_ID152">
                                                <SHORT-
NAME>DATASEG_CPU1_V01</SHORT-NAME>
                                                <SOURCE-START-
ADDRESS>00</SOURCE-START-ADDRESS>
                                                <COMPRESSED-
SIZE>15045</COMPRESSED-SIZE>
                                                <UNCOMPRESSED-
SIZE>16384</UNCOMPRESSED-SIZE>
                                            </SEGMENT>
                                        </SEGMENTS>
                                        <TARGET-ADDR-OFFSET xsi:type="POS-
OFFSET">
                                            <POSITIVE-OFFSET>010000</POSITIVE-
OFFSET>
                                        </TARGET-ADDR-OFFSET>
                                    </DATABLOCK>
                                    <DATABLOCK ID="FLASH_ECU_ID153" TYPE="CODE">
                                        <SHORT-NAME>CODE_CPU2_V01</SHORT-NAME>
                                        <FLASHDATA-REF ID-REF="FLASH_ECU_ID172"/>
```

```
                                    </DATABLOCK>
                                    <DATABLOCK ID="FLASH_ECU_ID154" TYPE="DATA">
                                          <SHORT-NAME>DATA_CPU2_V01</SHORT-NAME>
                                          <FLASHDATA-REF ID-REF="FLASH_ECU_ID173"/>
                                          <SEGMENTS>
                                                <SEGMENT ID="FLASH_ECU_ID155">
                                                      <SHORT-
NAME>DATASEG_CPU2_V01</SHORT-NAME>
                                                      <SOURCE-START-
ADDRESS>6000</SOURCE-START-ADDRESS>
                                                      <COMPRESSED-
SIZE>7830</COMPRESSED-SIZE>
                                                      <UNCOMPRESSED-
SIZE>8192</UNCOMPRESSED-SIZE>
                                                </SEGMENT>
                                          </SEGMENTS>
                                    </DATABLOCK>
                                    <DATABLOCK ID="FLASH_ECU_ID156" TYPE="CODE">
                                          <SHORT-NAME>CODE_CPU1_V02</SHORT-NAME>
                                          <FLASHDATA-REF ID-REF="FLASH_ECU_ID174"/>
                                    </DATABLOCK>
                                    <DATABLOCK ID="FLASH_ECU_ID157" TYPE="DATA">
                                          <SHORT-NAME>DATA_CPU1_V02</SHORT-NAME>
                                          <FLASHDATA-REF ID-REF="FLASH_ECU_ID175"/>
                                          <SEGMENTS>
                                                <SEGMENT ID="FLASH_ECU_ID158">
                                                      <SHORT-
NAME>DATASEG_CPU1_V02</SHORT-NAME>
                                                      <SOURCE-START-
ADDRESS>016384</SOURCE-START-ADDRESS>
                                                      <COMPRESSED-
SIZE>15344</COMPRESSED-SIZE>
                                                      <UNCOMPRESSED-
SIZE>10000</UNCOMPRESSED-SIZE>
                                                </SEGMENT>
                                          </SEGMENTS>
                                          <TARGET-ADDR-OFFSET xsi:type="POS-
OFFSET">
                                                <POSITIVE-OFFSET>010000</POSITIVE-
OFFSET>
                                          </TARGET-ADDR-OFFSET>
                                    </DATABLOCK>
                                    <DATABLOCK ID="FLASH_ECU_ID159" TYPE="CODE">
                                          <SHORT-NAME>CODE_CPU2_V02</SHORT-NAME>
                                          <FLASHDATA-REF ID-REF="FLASH_ECU_ID176"/>
                                    </DATABLOCK>
                                    <DATABLOCK ID="FLASH_ECU_ID160" TYPE="DATA">
                                          <SHORT-NAME>DATA_CPU2_V02</SHORT-NAME>
                                          <FLASHDATA-REF ID-REF="FLASH_ECU_ID177"/>
                                          <SEGMENTS>
                                                <SEGMENT ID="FLASH_ECU_ID161">
                                                      <SHORT-
NAME>DATASEG_CPU2_V02</SHORT-NAME>
                                                      <SOURCE-START-
ADDRESS>6000</SOURCE-START-ADDRESS>
                                                      <COMPRESSED-
SIZE>7703</COMPRESSED-SIZE>
                                                      <UNCOMPRESSED-
SIZE>8192</UNCOMPRESSED-SIZE>
                                                </SEGMENT>
                                          </SEGMENTS>
                                    </DATABLOCK>
                                    <DATABLOCK ID="FLASH_ECU_ID162" TYPE="CODE">
                                          <SHORT-NAME>CODE_CPU1_V02</SHORT-NAME>
                                          <FLASHDATA-REF ID-REF="FLASH_ECU_ID174"/>
                                          <FILTERS>
                                                <FILTER xsi:type="ADDRDEF-FILTER">
```

```
                                                        <FILTER-START>2000</FILTER-
START>
                                                        <FILTER-END>5FFF</FILTER-
END>
                                                </FILTER>
                                        </FILTERS>
                                        <SECURITYS>
                                                <SECURITY>
                                                        <SECURITY-METHOD
TYPE="A_ASCIISTRING">SECMETH_SIG01</SECURITY-METHOD>
                                                        <FW-SIGNATURE
TYPE="A_BYTEFIELD">1F62BA</FW-SIGNATURE>
                                                </SECURITY>
                                        </SECURITYS>
                                </DATABLOCK>
                                <DATABLOCK ID="FLASH_ECU_ID163" TYPE="DATA">
                                        <SHORT-NAME>DATA_CPU1_V02</SHORT-NAME>
                                        <FLASHDATA-REF ID-REF="FLASH_ECU_ID175"/>
                                        <FILTERS>
                                                <FILTER xsi:type="SIZEDEF-FILTER">
                                                        <FILTER-
START>011000</FILTER-START>
                                                        <FILTER-SIZE>1024</FILTER-
SIZE>
                                                </FILTER>
                                        </FILTERS>
                                        <SEGMENTS>
                                                <SEGMENT ID="FLASH_ECU_ID164">
                                                        <SHORT-
NAME>DATASEG_CPU1_V02</SHORT-NAME>
                                                        <SOURCE-START-
ADDRESS>010000</SOURCE-START-ADDRESS>
                                                        <SOURCE-END-
ADDRESS>013FFF</SOURCE-END-ADDRESS>
                                                </SEGMENT>
                                        </SEGMENTS>
                                </DATABLOCK>
                        </DATABLOCKS>
                        <FLASHDATAS>
                                <FLASHDATA ID="FLASH_ECU_ID170"
xsi:type="INTERN-FLASHDATA">
                                        <SHORT-NAME>FLASHDATA_CPU1_01</SHORT-
NAME>
                                        <DATAFORMAT SELECTION="INTEL-HEX"/>
                                        <DATA>
                                        :020000020000FC

      :100000003821F64FB80000003800BF193800000052

      :100010003821F64FB80000003821F64FB800000034
                                        ...
                                </DATA>
                                </FLASHDATA>
                                <FLASHDATA ID="FLASH_ECU_ID171"
xsi:type="INTERN-FLASHDATA">
                                        <SHORT-NAME>FLASHDATA_CPU1_02</SHORT-
NAME>
                                        <DATAFORMAT SELECTION="BINARY"/>
                                        <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">7F3F</ENCRYPT-COMPRESS-METHOD>
                                        <DATA>

      110300003821F64FB80000003800BF193800000052

      113010003821F64FB80000003821F64FB800000034

      113020003821F64FB80000003880BC81380000004D
```

```
                                    ...
                                </DATA>
                                </FLASHDATA>
                                <FLASHDATA ID="FLASH_ECU_ID172"
xsi:type="INTERN-FLASHDATA">
                                    <SHORT-NAME>FLASHDATA_CPU2_01</SHORT-
NAME>
                                    <DATAFORMAT SELECTION="MOTOROLA-S"/>
                                    <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
                                    <DATA>
                                    S020000020000FC

      S113000003821F64FB80000003800BF193800000052

      S113010003821F64FB80000003821F64FB800000034
                                    ...
                                </DATA>
                                </FLASHDATA>
                                <FLASHDATA ID="FLASH_ECU_ID173"
xsi:type="EXTERN-FLASHDATA">
                                    <SHORT-NAME>FLASHDATA_CPU2_02</SHORT-
NAME>
                                    <DATAFORMAT SELECTION="BINARY"/>
                                    <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
                                    <DATAFILE LATEBOUND-
DATAFILE="true">FLASHDATA_03.bin</DATAFILE>
                                </FLASHDATA>
                                <FLASHDATA ID="FLASH_ECU_ID174"
xsi:type="INTERN-FLASHDATA">
                                    <SHORT-NAME>FLASHDATA_CPU1_01</SHORT-
NAME>
                                    <DATAFORMAT SELECTION="INTEL-HEX"/>
                                    <DATA>
                                    :020000020000FC

      :100000003821F64FB80000003800BF193800000052

      :100010003821F64FB80000003821F64FB800000034
                                    ...
                                </DATA>
                                </FLASHDATA>
                                <FLASHDATA ID="FLASH_ECU_ID175"
xsi:type="INTERN-FLASHDATA">
                                    <SHORT-NAME>FLASHDATA_CPU1_02</SHORT-
NAME>
                                    <DATAFORMAT SELECTION="BINARY"/>
                                    <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">7F3F</ENCRYPT-COMPRESS-METHOD>
                                    <DATA>

      110300003821F64FB80000003800BF193800000052

      113010003821F64FB80000003821F64FB800000034

      113020003821F64FB80000003880BC81380000004D
                                    ...
                                </DATA>
                                </FLASHDATA>
                                <FLASHDATA ID="FLASH_ECU_ID176"
xsi:type="INTERN-FLASHDATA">
                                    <SHORT-NAME>FLASHDATA_CPU2_01</SHORT-
NAME>
                                    <DATAFORMAT SELECTION="MOTOROLA-S"/>
                                    <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
```

```
                                        <DATA>
                                        S020000020000FC

      S113000003821F64FB80000003800BF193800000052

      S113010003821F64FB80000003821F64FB800000034
                                        ...
                                  </DATA>
                                  </FLASHDATA>
                                  <FLASHDATA ID="FLASH_ECU_ID177"
xsi:type="EXTERN-FLASHDATA">
                                        <SHORT-NAME>FLASHDATA_CPU2_02</SHORT-
NAME>
                                        <DATAFORMAT SELECTION="BINARY"/>
                                        <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
                                        <DATAFILE LATEBOUND-
DATAFILE="true">FLASHDATA_07.bin</DATAFILE>
                                  </FLASHDATA>
                            </FLASHDATAS>
                      </MEM>
                      <PHYS-MEM ID="FLASH_ECU_ID190">
                            <SHORT-NAME/>
                            <PHYS-SEGMENTS>
                                  <PHYS-SEGMENT ID="FLASH_ECU_ID191"
xsi:type="ADDRDEF-PHYS-SEGMENT">
                                        <SHORT-NAME>CPU1_SEGMENT_01</SHORT-NAME>
                                        <FILLBYTE>FF</FILLBYTE>
                                        <BLOCK-SIZE>1024</BLOCK-SIZE>
                                        <START-ADDRESS>00</START-ADDRESS>
                                        <END-ADDRESS>FFFF</END-ADDRESS>
                                  </PHYS-SEGMENT>
                                  <PHYS-SEGMENT ID="FLASH_ECU_ID192"
xsi:type="ADDRDEF-PHYS-SEGMENT">
                                        <SHORT-NAME>CPU1_SEGMENT_02</SHORT-NAME>
                                        <FILLBYTE>FF</FILLBYTE>
                                        <BLOCK-SIZE>1024</BLOCK-SIZE>
                                        <START-ADDRESS>010000</START-ADDRESS>
                                        <END-ADDRESS>013FFF</END-ADDRESS>
                                  </PHYS-SEGMENT>
                                  <PHYS-SEGMENT ID="FLASH_ECU_ID193"
xsi:type="SIZEDEF-PHYS-SEGMENT">
                                        <SHORT-NAME>CPU2_SEGMENT_01</SHORT-NAME>
                                        <FILLBYTE>00</FILLBYTE>
                                        <BLOCK-SIZE>512</BLOCK-SIZE>
                                        <START-ADDRESS>00</START-ADDRESS>
                                        <SIZE>24576</SIZE>
                                  </PHYS-SEGMENT>
                                  <PHYS-SEGMENT ID="FLASH_ECU_ID194"
xsi:type="SIZEDEF-PHYS-SEGMENT">
                                        <SHORT-NAME>CPU2_SEGMENT_02</SHORT-NAME>
                                        <FILLBYTE>00</FILLBYTE>
                                        <BLOCK-SIZE>512</BLOCK-SIZE>
                                        <START-ADDRESS>6000</START-ADDRESS>
                                        <SIZE>8192</SIZE>
                                  </PHYS-SEGMENT>
                            </PHYS-SEGMENTS>
                      </PHYS-MEM>
                </ECU-MEM>
                </ECU-MEMS>
                <ECU-MEM-CONNECTORS>
                      <ECU-MEM-CONNECTOR ID="FLASH_ECU_ID230">
                            <SHORT-NAME>FLASH_ECU_Connector</SHORT-NAME>
                            <SESSION-DESCS>
                                  <SESSION-DESC DIRECTION="DOWNLOAD">
                                        <SHORT-
NAME>DOWNLOAD_SW_V01_COMPLETE</SHORT-NAME>
```

```
                                        <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                                        <PRIORITY>0</PRIORITY>
                                        <SESSION-SNREF SHORT-
NAME="FLASH_ECU_SW_V01_ON_HW_V01"/>
                                        <DIAG-COMM-SNREF SHORT-
NAME="DIAGCOMM_FLASH_JOB_01"/>
                                    </SESSION-DESC>
                                    <SESSION-DESC DIRECTION="DOWNLOAD">
                                        <SHORT-NAME>UPDATE_DATA_V02</SHORT-NAME>
                                        <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                                        <PRIORITY>50</PRIORITY>
                                        <SESSION-SNREF SHORT-
NAME="UPDATE_DATA_V02"/>
                                        <DIAG-COMM-SNREF SHORT-
NAME="DIAGCOMM_FLASH_JOB_02"/>
                                    </SESSION-DESC>
                                    <SESSION-DESC DIRECTION="DOWNLOAD">
                                        <SHORT-
NAME>UPDATE_CPU1_CODE_AND_DATA_V02</SHORT-NAME>
                                        <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                                        <PRIORITY>10</PRIORITY>
                                        <SESSION-SNREF SHORT-
NAME="UPDATE_CPU1_HW_V02_WITH_SW_V02"/>
                                        <DIAG-COMM-SNREF SHORT-
NAME="DIAGCOMM_FLASH_JOB_03"/>
                                    </SESSION-DESC>
                                    <SESSION-DESC DIRECTION="DOWNLOAD">
                                        <SHORT-
NAME>DOWNLOAD_SW_V02_COMPLETE</SHORT-NAME>
                                        <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                                        <PRIORITY>0</PRIORITY>
                                        <SESSION-SNREF SHORT-
NAME="FLASH_ECU_SW_V02_ON_HW_V02"/>
                                        <DIAG-COMM-SNREF SHORT-
NAME="DIAGCOMM_FLASH_ECU_JOB_01"/>
                                    </SESSION-DESC>
                            </SESSION-DESCS>
                            <ECU-MEM-REF ID-REF="FLASH_ECU_ID001"/>
                        </ECU-MEM-CONNECTOR>
                </ECU-MEM-CONNECTORS>
        </FLASH>
</ODX>
```

Annex D
(normative)

XML-Schema

D.1 XML Schema of ODX (odx.xsd)

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:include schemaLocation="odx-xhtml.xsd"/>
    <xsd:complexType abstract="false" name="ADDITIONAL-AUDIENCE">
        <!--Class: ADDITIONAL-AUDIENCE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="ADDITIONAL-AUDIENCES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="ADDITIONAL-
AUDIENCE" name="ADDITIONAL-AUDIENCE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ADDRDEF-FILTER">
        <!--Class: ADDRDEF-FILTER-->
        <xsd:complexContent>
            <xsd:extension base="FILTER">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="FILTER-END"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ADDRDEF-PHYS-SEGMENT">
        <!--Class: ADDRDEF-PHYS-SEGMENT-->
        <xsd:complexContent>
            <xsd:extension base="PHYS-SEGMENT">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="END-ADDRESS"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:simpleType name="ADDRESSING">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="FUNCTIONAL"/>
            <xsd:enumeration value="PHYSICAL"/>
            <xsd:enumeration value="FUNCTIONAL-OR-PHYSICAL"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="ADMIN-DATA">
        <!--Class: ADMIN-DATA-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:language"
name="LANGUAGE"/>
            <xsd:element type="COMPANY-DOC-INFOS" name="COMPANY-DOC-INFOS"
minOccurs="0" maxOccurs="1"/>
```

```
            <xsd:element type="DOC-REVISIONS" name="DOC-REVISIONS" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ALL-VALUE">
        <!--Class: ALL-VALUE-->
    </xsd:complexType>
    <xsd:complexType name="ALL-VARIANT-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="ALL-VARIANT-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ALLVARIANTS">
        <!--Class: ALLVARIANTS-->
    </xsd:complexType>
    <xsd:complexType abstract="false" name="AUDIENCE">
        <!--Class: AUDIENCE-->
        <xsd:choice>
            <xsd:element type="ENABLED-AUDIENCE-SNREFS" name="ENABLED-AUDIENCE-
SNREFS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="DISABLED-AUDIENCE-SNREFS" name="DISABLED-AUDIENCE-
SNREFS" minOccurs="0" maxOccurs="1"/>
        </xsd:choice>
        <xsd:attribute default="true" use="optional" type="xsd:boolean" name="IS-
SUPPLIER"/>
        <xsd:attribute default="true" use="optional" type="xsd:boolean" name="IS-
DEVELOPMENT"/>
        <xsd:attribute default="true" use="optional" type="xsd:boolean" name="IS-
MANUFACTURING"/>
        <xsd:attribute default="true" use="optional" type="xsd:boolean" name="IS-
AFTERSALES"/>
        <xsd:attribute default="true" use="optional" type="xsd:boolean" name="IS-
AFTERMARKET"/>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="BASE-COMPARAM">
        <!--Class: BASE-COMPARAM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="required" type="xsd:string" name="PARAM-CLASS"/>
        <xsd:attribute use="required" type="STANDARDISATION-LEVEL"
name="CPTYPE"/>
        <xsd:attribute use="required" type="xsd:unsignedInt" name="DISPLAY-
LEVEL"/>
        <xsd:attribute use="required" type="USAGE" name="CPUSAGE"/>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="BASE-FUNCTION-NODE">
        <!--Class: BASE-FUNCTION-NODE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="AUDIENCE"/>
            <xsd:element type="FUNCTION-IN-PARAMS" name="FUNCTION-IN-PARAMS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="FUNCTION-OUT-PARAMS" name="FUNCTION-OUT-PARAMS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="VEHICLE-INFORMATION-CONNECTORS" name="VEHICLE-
INFORMATION-CONNECTORS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="BASE-VALUE">
```

```
        <!--Class: BASE-VALUE-->
    </xsd:complexType>
    <xsd:complexType abstract="false" name="BASE-VARIANT">
        <!--Class: BASE-VARIANT-->
        <xsd:complexContent>
            <xsd:extension base="HIERARCHY-ELEMENT">
                <xsd:sequence>
                    <xsd:element type="DIAG-VARIABLES" name="DIAG-VARIABLES"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="VARIABLE-GROUPS" name="VARIABLE-GROUPS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="DYN-DEFINED-
SPEC" name="DYN-DEFINED-SPEC"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="BASE-VARIANT-
PATTERN" name="BASE-VARIANT-PATTERN"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="PARENT-REFS"
name="PARENT-REFS"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="BASE-VARIANT-PATTERN">
        <!--Class: BASE-VARIANT-PATTERN-->
        <xsd:sequence>
            <xsd:element type="MATCHING-BASE-VARIANT-PARAMETERS" name="MATCHING-
BASE-VARIANT-PARAMETERS" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="BASE-VARIANT-REF">
        <!--Class: BASE-VARIANT-REF-->
        <xsd:complexContent>
            <xsd:extension base="PARENT-REF"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="BASE-VARIANT-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="BASE-VARIANT-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="BASE-VARIANTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="BASE-VARIANT"
name="BASE-VARIANT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="BASIC-STRUCTURE">
        <!--Class: BASIC-STRUCTURE-->
        <xsd:complexContent>
            <xsd:extension base="COMPLEX-DOP">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="0"
type="xsd:unsignedInt" name="BYTE-SIZE"/>
                    <xsd:element type="PARAMS" name="PARAMS" minOccurs="1"
maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="CASE">
        <!--Class: CASE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="STRUCTURE-REF"
type="ODXLINK"/>
```

```
            <xsd:element maxOccurs="1" minOccurs="1" type="LIMIT" name="LOWER-
LIMIT"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="LIMIT" name="UPPER-
LIMIT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CASES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="CASE"
name="CASE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="CHECKSUM">
        <!--Class: CHECKSUM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:hexBinary"
name="FILLBYTE"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="SOURCE-START-ADDRESS"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:unsignedInt"
name="COMPRESSED-SIZE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="CHECKSUM-ALG"/>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" type="SOURCE-END-
ADDRESS" name="SOURCE-END-ADDRESS"/>
                <xsd:element maxOccurs="1" minOccurs="1" type="UNCOMPRESSED-SIZE"
name="UNCOMPRESSED-SIZE"/>
            </xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="1" type="CHECKSUM-RESULT"
name="CHECKSUM-RESULT"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="CHECKSUM-RESULT">
        <!--Class: CHECKSUM-RESULT-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="SESSION-SUB-ELEM-TYPE"
name="TYPE"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="CHECKSUMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="CHECKSUM"
name="CHECKSUM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="CODED-CONST">
        <!--Class: CODED-CONST-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="CODED-VALUE"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="DIAG-CODED-
TYPE" name="DIAG-CODED-TYPE"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CODED-VALUES">
```

```
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="xsd:string"
name="CODED-VALUE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMM-RELATION">
        <!--Class: COMM-RELATION-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION"
name="DESC"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="RELATION-TYPE"/>
            <xsd:choice maxOccurs="1" minOccurs="1">
                <xsd:element name="DIAG-COMM-REF" type="ODXLINK"/>
                <xsd:element name="DIAG-COMM-SNREF" type="SNREF"/>
            </xsd:choice>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="0" name="IN-PARAM-IF-SNREF"
type="SNREF"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="OUT-PARAM-IF-
SNREF" type="SNREF"/>
            </xsd:choice>
        </xsd:sequence>
        <xsd:attribute use="optional" type="COMM-RELATION-VALUE-TYPE"
name="VALUE-TYPE"/>
    </xsd:complexType>
    <xsd:complexType name="COMM-RELATIONS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="COMM-RELATION"
name="COMM-RELATION"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="COMM-RELATION-VALUE-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="CURRENT"/>
            <xsd:enumeration value="STORED"/>
            <xsd:enumeration value="STATIC"/>
            <xsd:enumeration value="SUBSTITUTED"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="COMPANY-DATA">
        <!--Class: COMPANY-DATA-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="ROLES" name="ROLES" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="TEAM-MEMBERS" name="TEAM-MEMBERS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="COMPANY-SPECIFIC-INFO"
name="COMPANY-SPECIFIC-INFO"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="COMPANY-DATAS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="COMPANY-DATA"
name="COMPANY-DATA"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPANY-DOC-INFO">
        <!--Class: COMPANY-DOC-INFO-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="COMPANY-DATA-REF"
type="ODXLINK"/>
```

```
            <xsd:element maxOccurs="1" minOccurs="0" name="TEAM-MEMBER-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string" name="DOC-
LABEL"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="COMPANY-DOC-INFOS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="COMPANY-DOC-
INFO" name="COMPANY-DOC-INFO"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPANY-REVISION-INFO">
        <!--Class: COMPANY-REVISION-INFO-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="COMPANY-DATA-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="REVISION-LABEL"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="STATE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="COMPANY-REVISION-INFOS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="COMPANY-
REVISION-INFO" name="COMPANY-REVISION-INFO"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPANY-SPECIFIC-INFO">
        <!--Class: COMPANY-SPECIFIC-INFO-->
        <xsd:sequence>
            <xsd:element type="RELATED-DOCS" name="RELATED-DOCS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPARAM">
        <!--Class: COMPARAM-->
        <xsd:complexContent>
            <xsd:extension base="BASE-COMPARAM">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="PHYSICAL-DEFAULT-VALUE"/>
                    <xsd:element maxOccurs="1" minOccurs="1" name="DATA-OBJECT-
PROP-REF" type="ODXLINK"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPARAM-REF">
        <!--Class: COMPARAM-REF-->
        <xsd:sequence>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" type="SIMPLE-VALUE"
name="SIMPLE-VALUE"/>
                <xsd:element maxOccurs="1" minOccurs="1" type="COMPLEX-VALUE"
name="COMPLEX-VALUE"/>
            </xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION"
name="DESC"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="PROTOCOL-SNREF"
type="SNREF"/>
```

```
            <xsd:element maxOccurs="1" minOccurs="0" name="PROT-STACK-SNREF"
type="SNREF"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="ODXLINK-ATTR"/>
    </xsd:complexType>
    <xsd:complexType name="COMPARAM-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="COMPARAM-REF"
name="COMPARAM-REF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="COMPARAMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="COMPARAM"
name="COMPARAM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPARAM-SPEC">
        <!--Class: COMPARAM-SPEC-->
        <xsd:complexContent>
            <xsd:extension base="ODX-CATEGORY">
                <xsd:sequence>
                    <xsd:element type="PROT-STACKS" name="PROT-STACKS"
minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPARAM-SUBSET">
        <!--Class: COMPARAM-SUBSET-->
        <xsd:complexContent>
            <xsd:extension base="ODX-CATEGORY">
                <xsd:sequence>
                    <xsd:element type="COMPARAMS" name="COMPARAMS" minOccurs="0"
maxOccurs="1"/>
                    <xsd:element type="COMPLEX-COMPARAMS" name="COMPLEX-
COMPARAMS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="DATA-OBJECT-PROPS" name="DATA-OBJECT-
PROPS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="UNIT-SPEC"
name="UNIT-SPEC"/>
                </xsd:sequence>
                <xsd:attribute use="required" type="xsd:string" name="CATEGORY"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="COMPARAM-SUBSET-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="COMPARAM-
SUBSET-REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPLEX-COMPARAM">
        <!--Class: COMPLEX-COMPARAM-->
        <xsd:complexContent>
            <xsd:extension base="BASE-COMPARAM">
                <xsd:choice maxOccurs="unbounded" minOccurs="1">
                    <xsd:element maxOccurs="1" minOccurs="1" type="COMPARAM"
name="COMPARAM"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="COMPLEX-
COMPARAM" name="COMPLEX-COMPARAM"/>
                </xsd:choice>
                <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="ALLOW-MULTIPLE-VALUES"/>
```

```
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="COMPLEX-COMPARAMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="COMPLEX-
COMPARAM" name="COMPLEX-COMPARAM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="COMPLEX-DOP">
        <!--Class: COMPLEX-DOP-->
        <xsd:complexContent>
            <xsd:extension base="DOP-BASE"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPLEX-VALUE">
        <!--Class: COMPLEX-VALUE-->
        <xsd:complexContent>
            <xsd:extension base="BASE-VALUE">
                <xsd:choice maxOccurs="unbounded" minOccurs="1">
                    <xsd:element maxOccurs="1" minOccurs="1" type="SIMPLE-VALUE"
name="SIMPLE-VALUE"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="COMPLEX-VALUE"
name="COMPLEX-VALUE"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:simpleType name="COMPU-CATEGORY">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="IDENTICAL"/>
            <xsd:enumeration value="LINEAR"/>
            <xsd:enumeration value="SCALE-LINEAR"/>
            <xsd:enumeration value="TEXTTABLE"/>
            <xsd:enumeration value="COMPUCODE"/>
            <xsd:enumeration value="TAB-INTP"/>
            <xsd:enumeration value="RAT-FUNC"/>
            <xsd:enumeration value="SCALE-RAT-FUNC"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="COMPU-CONST">
        <!--Class: COMPU-CONST-->
        <xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="1" type="V" name="V"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="VT" name="VT"/>
        </xsd:choice>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPU-DEFAULT-VALUE">
        <!--Class: COMPU-DEFAULT-VALUE-->
        <xsd:sequence>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" type="V" name="V"/>
                <xsd:element maxOccurs="1" minOccurs="1" type="VT" name="VT"/>
            </xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="0" type="COMPU-INVERSE-VALUE"
name="COMPU-INVERSE-VALUE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="COMPU-DENOMINATOR">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="V" name="V"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPU-INTERNAL-TO-PHYS">
        <!--Class: COMPU-INTERNAL-TO-PHYS-->
```

```
        <xsd:sequence>
            <xsd:choice>
                <xsd:element type="COMPU-SCALES" name="COMPU-SCALES"
minOccurs="1" maxOccurs="1"/>
                <xsd:element maxOccurs="1" minOccurs="1" type="PROG-CODE"
name="PROG-CODE"/>
            </xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="0" type="COMPU-DEFAULT-VALUE"
name="COMPU-DEFAULT-VALUE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPU-INVERSE-VALUE">
        <!--Class: COMPU-INVERSE-VALUE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="V" name="V"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPU-METHOD">
        <!--Class: COMPU-METHOD-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="COMPU-CATEGORY"
name="CATEGORY"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="COMPU-INTERNAL-TO-
PHYS" name="COMPU-INTERNAL-TO-PHYS"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="COMPU-PHYS-TO-
INTERNAL" name="COMPU-PHYS-TO-INTERNAL"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="COMPU-NUMERATOR">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="V" name="V"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPU-PHYS-TO-INTERNAL">
        <!--Class: COMPU-PHYS-TO-INTERNAL-->
        <xsd:sequence>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" type="PROG-CODE"
name="PROG-CODE"/>
                <xsd:element type="COMPU-SCALES" name="COMPU-SCALES"
minOccurs="1" maxOccurs="1"/>
            </xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="0" type="COMPU-DEFAULT-VALUE"
name="COMPU-DEFAULT-VALUE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPU-RATIONAL-COEFFS">
        <!--Class: COMPU-RATIONAL-COEFFS-->
        <xsd:sequence>
            <xsd:element type="COMPU-NUMERATOR" name="COMPU-NUMERATOR"
minOccurs="1" maxOccurs="1"/>
            <xsd:element type="COMPU-DENOMINATOR" name="COMPU-DENOMINATOR"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="COMPU-SCALE">
        <!--Class: COMPU-SCALE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="TEXT" name="SHORT-
LABEL"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION"
name="DESC"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="LIMIT" name="LOWER-
LIMIT"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="LIMIT" name="UPPER-
LIMIT"/>
```

```
            <xsd:element maxOccurs="1" minOccurs="0" type="COMPU-INVERSE-VALUE"
name="COMPU-INVERSE-VALUE"/>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" type="COMPU-CONST"
name="COMPU-CONST"/>
                <xsd:element maxOccurs="1" minOccurs="1" type="COMPU-RATIONAL-
COEFFS" name="COMPU-RATIONAL-COEFFS"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="COMPU-SCALES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="COMPU-SCALE"
name="COMPU-SCALE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="CONFIG-DATA">
        <!--Class: CONFIG-DATA-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="VALID-BASE-VARIANTS" name="VALID-BASE-VARIANTS"
minOccurs="1" maxOccurs="1"/>
            <xsd:element type="CONFIG-RECORDS" name="CONFIG-RECORDS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="CONFIG-DATA-DICTIONARY-SPEC">
        <!--Class: CONFIG-DATA-DICTIONARY-SPEC-->
        <xsd:sequence>
            <xsd:element type="DATA-OBJECT-PROPS" name="DATA-OBJECT-PROPS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="UNIT-SPEC" name="UNIT-
SPEC"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CONFIG-DATAS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="CONFIG-DATA"
name="CONFIG-DATA"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="CONFIG-ID-ITEM">
        <!--Class: CONFIG-ID-ITEM-->
        <xsd:complexContent>
            <xsd:extension base="CONFIG-ITEM"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="CONFIG-ITEM">
        <!--Class: CONFIG-ITEM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:unsignedInt"
name="BYTE-POSITION"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:unsignedInt"
name="BIT-POSITION"/>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-ROW-REF"
type="ODXLINK"/>
                <xsd:group maxOccurs="1" minOccurs="1" ref="ODXLINK-TO-TABLE2"/>
                <xsd:group maxOccurs="1" minOccurs="1" ref="SNREF-TO-TABLE2"/>
                <xsd:element maxOccurs="1" minOccurs="1" name="DATA-OBJECT-PROP-
REF" type="ODXLINK"/>
                <xsd:element maxOccurs="1" minOccurs="1" name="DATA-OBJECT-PROP-
SNREF" type="SNREF"/>
```

```
            </xsd:choice>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="SEMANTIC"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="CONFIG-RECORD">
        <!--Class: CONFIG-RECORD-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="CONFIG-ID-ITEM"
name="CONFIG-ID-ITEM"/>
            <xsd:element type="DIAG-COMM-DATA-CONNECTORS" name="DIAG-COMM-DATA-
CONNECTORS" minOccurs="1" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="IDENT-VALUE"
name="CONFIG-ID"/>
            <xsd:element type="DATA-RECORDS" name="DATA-RECORDS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="AUDIENCE"/>
            <xsd:element type="SYSTEM-ITEMS" name="SYSTEM-ITEMS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="DATA-ID-ITEM"
name="DATA-ID-ITEM"/>
            <xsd:element type="OPTION-ITEMS" name="OPTION-ITEMS" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CONFIG-RECORDS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="CONFIG-RECORD"
name="CONFIG-RECORD"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DATA">
        <!--Class: DATA-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DATABLOCK">
        <!--Class: DATABLOCK-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="FLASHDATA-REF"
type="ODXLINK"/>
            <xsd:element type="FILTERS" name="FILTERS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="SEGMENTS" name="SEGMENTS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="TARGET-ADDR-OFFSET"
name="TARGET-ADDR-OFFSET"/>
            <xsd:element type="OWN-IDENTS" name="OWN-IDENTS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="SECURITYS" name="SECURITYS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="AUDIENCE"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="required" type="xsd:string" name="TYPE"/>
    </xsd:complexType>
    <xsd:complexType name="DATABLOCK-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
```

```
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="DATABLOCK-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DATABLOCKS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DATABLOCK"
name="DATABLOCK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DATAFILE">
        <!--Class: DATAFILE-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="xsd:boolean" name="LATEBOUND-
DATAFILE"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DATAFORMAT">
        <!--Class: DATAFORMAT-->
        <xsd:attribute use="required" type="DATAFORMAT-SELECTION"
name="SELECTION"/>
    </xsd:complexType>
    <xsd:simpleType name="DATAFORMAT-SELECTION">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="INTEL-HEX"/>
            <xsd:enumeration value="MOTOROLA-S"/>
            <xsd:enumeration value="BINARY"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="DATA-ID-ITEM">
        <!--Class: DATA-ID-ITEM-->
        <xsd:complexContent>
            <xsd:extension base="CONFIG-ITEM"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DATA-OBJECT-PROP">
        <!--Class: DATA-OBJECT-PROP-->
        <xsd:complexContent>
            <xsd:extension base="DOP-BASE">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="COMPU-METHOD"
name="COMPU-METHOD"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="DIAG-CODED-
TYPE" name="DIAG-CODED-TYPE"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="PHYSICAL-TYPE"
name="PHYSICAL-TYPE"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="INTERNAL-
CONSTR" name="INTERNAL-CONSTR"/>
                    <xsd:element maxOccurs="1" minOccurs="0" name="UNIT-REF"
type="ODXLINK"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="INTERNAL-
CONSTR" name="PHYS-CONSTR"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="DATA-OBJECT-PROPS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DATA-OBJECT-
PROP" name="DATA-OBJECT-PROP"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DATA-RECORD">
```

```
        <!--Class: DATA-RECORD-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="DATAFORMAT-SELECTION"
name="DATAFORMAT"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="RULE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="KEY"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="IDENT-VALUE"
name="DATA-ID"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="0" type="DATAFILE"
name="DATAFILE"/>
                <xsd:element maxOccurs="1" minOccurs="0" type="DATA"
name="DATA"/>
            </xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="AUDIENCE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DATA-RECORDS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DATA-RECORD"
name="DATA-RECORD"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="DATA-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="A_INT32"/>
            <xsd:enumeration value="A_UINT32"/>
            <xsd:enumeration value="A_FLOAT32"/>
            <xsd:enumeration value="A_FLOAT64"/>
            <xsd:enumeration value="A_ASCIISTRING"/>
            <xsd:enumeration value="A_UTF8STRING"/>
            <xsd:enumeration value="A_UNICODE2STRING"/>
            <xsd:enumeration value="A_BYTEFIELD"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="DEFAULT-CASE">
        <!--Class: DEFAULT-CASE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="STRUCTURE-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DESCRIPTION">
        <!--Class: DESCRIPTION-->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" type="p" name="p"/>
            <xsd:element type="EXTERNAL-DOCS" name="EXTERNAL-DOCS" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="TI"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DETERMINE-NUMBER-OF-ITEMS">
        <!--Class: DETERMINE-NUMBER-OF-ITEMS-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:unsignedInt"
name="BYTE-POSITION"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="BIT-POSITION">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedInt">
                        <xsd:maxExclusive value="8"/>
```

```
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element maxOccurs="1" minOccurs="1" name="DATA-OBJECT-PROP-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="DIAG-CLASS-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="STARTCOMM"/>
            <xsd:enumeration value="STOPCOMM"/>
            <xsd:enumeration value="VARIANTIDENTIFICATION"/>
            <xsd:enumeration value="READ-DYN-DEFINED-MESSAGE"/>
            <xsd:enumeration value="DYN-DEF-MESSAGE"/>
            <xsd:enumeration value="CLEAR-DYN-DEF-MESSAGE"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="true" name="DIAG-CODED-TYPE">
        <!--Class: DIAG-CODED-TYPE-->
        <xsd:attribute use="optional" type="ENCODING" name="BASE-TYPE-ENCODING"/>
        <xsd:attribute use="required" type="DATA-TYPE" name="BASE-DATA-TYPE"/>
        <xsd:attribute default="true" use="optional" type="xsd:boolean" name="IS-
HIGHLOW-BYTE-ORDER"/>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="DIAG-COMM">
        <!--Class: DIAG-COMM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="FUNCT-CLASS-REFS" name="FUNCT-CLASS-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="AUDIENCE"/>
            <xsd:element type="PROTOCOL-SNREFS" name="PROTOCOL-SNREFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="RELATED-DIAG-COMM-REFS" name="RELATED-DIAG-COMM-
REFS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="PRE-CONDITION-STATE-REFS" name="PRE-CONDITION-
STATE-REFS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="STATE-TRANSITION-REFS" name="STATE-TRANSITION-
REFS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="optional" type="xsd:string" name="SEMANTIC"/>
        <xsd:attribute use="optional" type="DIAG-CLASS-TYPE" name="DIAGNOSTIC-
CLASS"/>
        <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-MANDATORY"/>
        <xsd:attribute default="true" use="optional" type="xsd:boolean" name="IS-
EXECUTABLE"/>
        <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-FINAL"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DIAG-COMM-DATA-CONNECTOR">
        <!--Class: DIAG-COMM-DATA-CONNECTOR-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:unsignedInt"
name="UNCOMPRESSED-SIZE"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="SOURCE-START-ADDRESS"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="READ-DIAG-COMM-
CONNECTOR" name="READ-DIAG-COMM-CONNECTOR"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="WRITE-DIAG-COMM-
CONNECTOR" name="WRITE-DIAG-COMM-CONNECTOR"/>
```

```
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DIAG-COMM-DATA-CONNECTORS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DIAG-COMM-
DATA-CONNECTOR" name="DIAG-COMM-DATA-CONNECTOR"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DIAG-COMM-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="DIAG-COMM-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DIAG-COMMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:group maxOccurs="unbounded" minOccurs="1" ref="DIAG-COMM-
PROXY"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DIAG-DATA-DICTIONARY-SPEC">
        <!--Class: DIAG-DATA-DICTIONARY-SPEC-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element type="DTC-DOPS" name="DTC-DOPS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="ENV-DATA-DESCS" name="ENV-DATA-DESCS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="DATA-OBJECT-PROPS" name="DATA-OBJECT-PROPS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="STRUCTURES" name="STRUCTURES" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="STATIC-FIELDS" name="STATIC-FIELDS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="DYNAMIC-LENGTH-FIELDS" name="DYNAMIC-LENGTH-
FIELDS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="DYNAMIC-ENDMARKER-FIELDS" name="DYNAMIC-ENDMARKER-
FIELDS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="END-OF-PDU-FIELDS" name="END-OF-PDU-FIELDS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="MUXS" name="MUXS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="ENV-DATAS" name="ENV-DATAS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="UNIT-SPEC" name="UNIT-
SPEC"/>
            <xsd:element type="TABLES" name="TABLES" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="DIAG-LAYER">
        <!--Class: DIAG-LAYER-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element type="COMPANY-DATAS" name="COMPANY-DATAS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="FUNCT-CLASSS" name="FUNCT-CLASSS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="DIAG-DATA-DICTIONARY-
SPEC" name="DIAG-DATA-DICTIONARY-SPEC"/>
            <xsd:element type="DIAG-COMMS" name="DIAG-COMMS" minOccurs="0"
maxOccurs="1"/>
```

```
            <xsd:element type="REQUESTS" name="REQUESTS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="POS-RESPONSES" name="POS-RESPONSES" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="NEG-RESPONSES" name="NEG-RESPONSES" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="GLOBAL-NEG-RESPONSES" name="GLOBAL-NEG-RESPONSES"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="IMPORT-REFS" name="IMPORT-REFS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="STATE-CHARTS" name="STATE-CHARTS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="ADDITIONAL-AUDIENCES" name="ADDITIONAL-AUDIENCES"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="SUB-COMPONENTS" name="SUB-COMPONENTS"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DIAG-LAYER-CONTAINER">
        <!--Class: DIAG-LAYER-CONTAINER-->
        <xsd:complexContent>
            <xsd:extension base="ODX-CATEGORY">
                <xsd:sequence>
                    <xsd:element type="PROTOCOLS" name="PROTOCOLS" minOccurs="0"
maxOccurs="1"/>
                    <xsd:element type="FUNCTIONAL-GROUPS" name="FUNCTIONAL-
GROUPS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="ECU-SHARED-DATAS" name="ECU-SHARED-DATAS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="BASE-VARIANTS" name="BASE-VARIANTS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="ECU-VARIANTS" name="ECU-VARIANTS"
minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="DIAG-LAYER-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="DIAG-LAYER-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DIAG-OBJECT-CONNECTOR">
        <!--Class: DIAG-OBJECT-CONNECTOR-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="EXECUTABLES" name="EXECUTABLES" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="ENV-DATA-REFS" name="ENV-DATA-REFS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="TABLE-REFS" name="TABLE-REFS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="TABLE-ROW-REFS" name="TABLE-ROW-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="DOP-BASE-REFS" name="DOP-BASE-REFS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="DTC-REFS" name="DTC-REFS" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DIAG-SERVICE">
```

```
        <!--Class: DIAG-SERVICE-->
        <xsd:complexContent>
            <xsd:extension base="DIAG-COMM">
                <xsd:sequence>
                    <xsd:element type="COMPARAM-REFS" name="COMPARAM-REFS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" name="REQUEST-REF"
type="ODXLINK"/>
                    <xsd:element type="POS-RESPONSE-REFS" name="POS-RESPONSE-
REFS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="NEG-RESPONSE-REFS" name="NEG-RESPONSE-
REFS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="POS-RESPONSE-
SUPPRESSABLE" name="POS-RESPONSE-SUPPRESSABLE"/>
                </xsd:sequence>
                <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-CYCLIC"/>
                <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-MULTIPLE"/>
                <xsd:attribute default="PHYSICAL" use="optional"
type="ADDRESSING" name="ADDRESSING"/>
                <xsd:attribute default="SEND-AND-RECEIVE" use="optional"
type="TRANS-MODE" name="TRANSMISSION-MODE"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DIAG-VARIABLE">
        <!--Class: DIAG-VARIABLE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="VARIABLE-GROUP-REF"
type="ODXLINK"/>
            <xsd:element type="SW-VARIABLES" name="SW-VARIABLES" minOccurs="0"
maxOccurs="1"/>
            <xsd:choice>
                <xsd:element type="COMM-RELATIONS" name="COMM-RELATIONS"
minOccurs="0" maxOccurs="1"/>
                <xsd:element maxOccurs="1" minOccurs="0" type="SNREF-TO-TABLEROW"
name="SNREF-TO-TABLEROW"/>
            </xsd:choice>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-READ-BEFORE-WRITE"/>
    </xsd:complexType>
    <xsd:complexType name="DIAG-VARIABLE-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="DIAG-VARIABLE-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DIAG-VARIABLES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:group maxOccurs="unbounded" minOccurs="1" ref="DIAG-VARIABLE-
PROXY"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="DIRECTION">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="DOWNLOAD"/>
            <xsd:enumeration value="UPLOAD"/>
        </xsd:restriction>
```

```
    </xsd:simpleType>
    <xsd:complexType name="DISABLED-AUDIENCE-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="DISABLED-
AUDIENCE-SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DOC-REVISION">
        <!--Class: DOC-REVISION-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" name="TEAM-MEMBER-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="REVISION-LABEL"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="STATE"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:dateTime"
name="DATE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="TOOL"/>
            <xsd:element type="COMPANY-REVISION-INFOS" name="COMPANY-REVISION-
INFOS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="MODIFICATIONS" name="MODIFICATIONS" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DOC-REVISIONS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DOC-REVISION"
name="DOC-REVISION"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="DOCTYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="FLASH"/>
            <xsd:enumeration value="CONTAINER"/>
            <xsd:enumeration value="LAYER"/>
            <xsd:enumeration value="MULTIPLE-ECU-JOB-SPEC"/>
            <xsd:enumeration value="COMPARAM-SPEC"/>
            <xsd:enumeration value="VEHICLE-INFO-SPEC"/>
            <xsd:enumeration value="COMPARAM-SUBSET"/>
            <xsd:enumeration value="ECU-CONFIG"/>
            <xsd:enumeration value="FUNCTION-DICTIONARY-SPEC"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="true" name="DOP-BASE">
        <!--Class: DOP-BASE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="DOP-BASE-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="DOP-BASE-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DOP-BASE-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
```

```
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="DOP-BASE-
SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DTC">
        <!--Class: DTC-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="SHORT-NAME">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:pattern value="[a-zA-Z0-9_]{1,128}"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:int"
name="TROUBLE-CODE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="DISPLAY-TROUBLE-CODE"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="TEXT" name="TEXT"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:unsignedByte"
name="LEVEL"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-TEMPORARY"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DTC-DOP">
        <!--Class: DTC-DOP-->
        <xsd:complexContent>
            <xsd:extension base="DOP-BASE">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="DIAG-CODED-
TYPE" name="DIAG-CODED-TYPE"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="PHYSICAL-TYPE"
name="PHYSICAL-TYPE"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="COMPU-METHOD"
name="COMPU-METHOD"/>
                    <xsd:element type="DTCS" name="DTCS" minOccurs="1"
maxOccurs="1"/>
                    <xsd:element type="LINKED-DTC-DOPS" name="LINKED-DTC-DOPS"
minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
                <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-VISIBLE"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="DTC-DOPS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DTC-DOP"
name="DTC-DOP"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DTC-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="DTC-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DTCS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:group maxOccurs="unbounded" minOccurs="1" ref="DTC-PROXY"/>
```

```
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DTC-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="DTC-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DTC-VALUE">
        <!--Class: DTC-VALUE-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:int"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="DTC-VALUES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DTC-VALUE"
name="DTC-VALUE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DYNAMIC">
        <!--Class: DYNAMIC-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DYNAMIC-ENDMARKER-FIELD">
        <!--Class: DYNAMIC-ENDMARKER-FIELD-->
        <xsd:complexContent>
            <xsd:extension base="FIELD">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="DYN-END-DOP-
REF" name="DATA-OBJECT-PROP-REF"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="DYNAMIC-ENDMARKER-FIELDS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DYNAMIC-
ENDMARKER-FIELD" name="DYNAMIC-ENDMARKER-FIELD"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DYNAMIC-LENGTH-FIELD">
        <!--Class: DYNAMIC-LENGTH-FIELD-->
        <xsd:complexContent>
            <xsd:extension base="FIELD">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1"
type="xsd:unsignedInt" name="OFFSET"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="DETERMINE-
NUMBER-OF-ITEMS" name="DETERMINE-NUMBER-OF-ITEMS"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="DYNAMIC-LENGTH-FIELDS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DYNAMIC-
LENGTH-FIELD" name="DYNAMIC-LENGTH-FIELD"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DYN-DEFINED-SPEC">
```

```
        <!--Class: DYN-DEFINED-SPEC-->
        <xsd:sequence>
            <xsd:element type="DYN-ID-DEF-MODE-INFOS" name="DYN-ID-DEF-MODE-
INFOS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DYN-END-DOP-REF">
        <!--Association Class: DYN-END-DOP-REF-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="TERMINATION-VALUE"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="ODXLINK-ATTR"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="DYN-ID-DEF-MODE-INFO">
        <!--Class: DYN-ID-DEF-MODE-INFO-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string" name="DEF-
MODE"/>
            <xsd:choice maxOccurs="1" minOccurs="1">
                <xsd:element name="CLEAR-DYN-DEF-MESSAGE-REF" type="ODXLINK"/>
                <xsd:element name="CLEAR-DYN-DEF-MESSAGE-SNREF" type="SNREF"/>
            </xsd:choice>
            <xsd:choice maxOccurs="1" minOccurs="1">
                <xsd:element name="READ-DYN-DEF-MESSAGE-REF" type="ODXLINK"/>
                <xsd:element name="READ-DYN-DEF-MESSAGE-SNREF" type="SNREF"/>
            </xsd:choice>
            <xsd:choice maxOccurs="1" minOccurs="1">
                <xsd:element name="DYN-DEF-MESSAGE-REF" type="ODXLINK"/>
                <xsd:element name="DYN-DEF-MESSAGE-SNREF" type="SNREF"/>
            </xsd:choice>
            <xsd:element type="SUPPORTED-DYN-IDS" name="SUPPORTED-DYN-IDS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="SELECTION-TABLE-REFS" name="SELECTION-TABLE-REFS"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DYN-ID-DEF-MODE-INFOS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="DYN-ID-DEF-
MODE-INFO" name="DYN-ID-DEF-MODE-INFO"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ECU-CONFIG">
        <!--Class: ECU-CONFIG-->
        <xsd:complexContent>
            <xsd:extension base="ODX-CATEGORY">
                <xsd:sequence>
                    <xsd:element type="CONFIG-DATAS" name="CONFIG-DATAS"
minOccurs="1" maxOccurs="1"/>
                    <xsd:element type="ADDITIONAL-AUDIENCES" name="ADDITIONAL-
AUDIENCES" minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="CONFIG-DATA-
DICTIONARY-SPEC" name="CONFIG-DATA-DICTIONARY-SPEC"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ECU-GROUP">
        <!--Class: ECU-GROUP-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="GROUP-MEMBERS" name="GROUP-MEMBERS" minOccurs="1"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
```

```
    </xsd:complexType>
    <xsd:complexType name="ECU-GROUPS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="ECU-GROUP"
name="ECU-GROUP"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ECU-MEM">
        <!--Class: ECU-MEM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="MEM" name="MEM"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="PHYS-MEM" name="PHYS-
MEM"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ECU-MEM-CONNECTOR">
        <!--Class: ECU-MEM-CONNECTOR-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element type="FLASH-CLASSS" name="FLASH-CLASSS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="SESSION-DESCS" name="SESSION-DESCS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="IDENT-DESCS" name="IDENT-DESCS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="ECU-MEM-REF"
type="ODXLINK"/>
            <xsd:element type="LAYER-REFS" name="LAYER-REFS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="ALL-VARIANT-REFS" name="ALL-VARIANT-REFS"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="ECU-MEM-CONNECTORS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="ECU-MEM-
CONNECTOR" name="ECU-MEM-CONNECTOR"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ECU-MEMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="ECU-MEM"
name="ECU-MEM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ECU-PROXY">
        <!--Class: ECU-PROXY-->
        <xsd:complexContent>
            <xsd:extension base="INFO-COMPONENT"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="ECU-PROXY-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
```

```
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="ECU-PROXY-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ECU-SHARED-DATA">
        <!--Class: ECU-SHARED-DATA-->
        <xsd:complexContent>
            <xsd:extension base="DIAG-LAYER">
                <xsd:sequence>
                    <xsd:element type="DIAG-VARIABLES" name="DIAG-VARIABLES"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="VARIABLE-GROUPS" name="VARIABLE-GROUPS"
minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ECU-SHARED-DATA-REF">
        <!--Class: ECU-SHARED-DATA-REF-->
        <xsd:complexContent>
            <xsd:extension base="PARENT-REF"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="ECU-SHARED-DATAS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="ECU-SHARED-
DATA" name="ECU-SHARED-DATA"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ECU-VARIANT">
        <!--Class: ECU-VARIANT-->
        <xsd:complexContent>
            <xsd:extension base="HIERARCHY-ELEMENT">
                <xsd:sequence>
                    <xsd:element type="DIAG-VARIABLES" name="DIAG-VARIABLES"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="VARIABLE-GROUPS" name="VARIABLE-GROUPS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="ECU-VARIANT-PATTERNS" name="ECU-VARIANT-
PATTERNS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="DYN-DEFINED-
SPEC" name="DYN-DEFINED-SPEC"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="PARENT-REFS"
name="PARENT-REFS"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ECU-VARIANT-PATTERN">
        <!--Class: ECU-VARIANT-PATTERN-->
        <xsd:sequence>
            <xsd:element type="MATCHING-PARAMETERS" name="MATCHING-PARAMETERS"
minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ECU-VARIANT-PATTERNS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="ECU-VARIANT-
PATTERN" name="ECU-VARIANT-PATTERN"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ECU-VARIANT-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
```

```
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="ECU-VARIANT-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ECU-VARIANTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="ECU-VARIANT"
name="ECU-VARIANT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ECU-VARIANT-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="ECU-VARIANT-
SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ENABLED-AUDIENCE-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="ENABLED-
AUDIENCE-SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="ENCODING">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="BCD-P"/>
            <xsd:enumeration value="BCD-UP"/>
            <xsd:enumeration value="1C"/>
            <xsd:enumeration value="2C"/>
            <xsd:enumeration value="SM"/>
            <xsd:enumeration value="UTF-8"/>
            <xsd:enumeration value="UCS-2"/>
            <xsd:enumeration value="ISO-8859-1"/>
            <xsd:enumeration value="ISO-8859-2"/>
            <xsd:enumeration value="WINDOWS-1252"/>
            <xsd:enumeration value="NONE"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="ENCRYPT-COMPRESS-METHOD">
        <!--Class: ENCRYPT-COMPRESS-METHOD-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="ENCRYPT-COMPRESS-METHOD-TYPE"
name="TYPE"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:simpleType name="ENCRYPT-COMPRESS-METHOD-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="A_UINT32"/>
            <xsd:enumeration value="A_BYTEFIELD"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="END-OF-PDU-FIELD">
        <!--Class: END-OF-PDU-FIELD-->
        <xsd:complexContent>
            <xsd:extension base="FIELD">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="0"
type="xsd:unsignedInt" name="MAX-NUMBER-OF-ITEMS"/>
                    <xsd:element maxOccurs="1" minOccurs="0"
type="xsd:unsignedInt" name="MIN-NUMBER-OF-ITEMS"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
```

```
    </xsd:complexType>
    <xsd:complexType name="END-OF-PDU-FIELDS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="END-OF-PDU-
FIELD" name="END-OF-PDU-FIELD"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ENV-DATA">
        <!--Class: ENV-DATA-->
        <xsd:complexContent>
            <xsd:extension base="BASIC-STRUCTURE">
                <xsd:choice>
                    <xsd:element maxOccurs="1" minOccurs="1" type="ALL-VALUE"
name="ALL-VALUE"/>
                    <xsd:element type="DTC-VALUES" name="DTC-VALUES"
minOccurs="0" maxOccurs="1"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ENV-DATA-DESC">
        <!--Class: ENV-DATA-DESC-->
        <xsd:complexContent>
            <xsd:extension base="COMPLEX-DOP">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="PARAM-SNREF"
type="SNREF"/>
                    <xsd:element type="ENV-DATAS" name="ENV-DATAS" minOccurs="1"
maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="ENV-DATA-DESCS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="ENV-DATA-DESC"
name="ENV-DATA-DESC"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ENV-DATA-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="ENV-DATA-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ENV-DATAS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:group maxOccurs="unbounded" minOccurs="1" ref="ENV-DATA-PROXY"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ENV-DATA-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="ENV-DATA-
SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="EXECUTABLE">
        <!--Class: EXECUTABLE-->
        <xsd:choice>
            <xsd:element type="DIAG-VARIABLE-REFS" name="DIAG-VARIABLE-REFS"
minOccurs="0" maxOccurs="1"/>
```

```
            <xsd:element type="DIAG-COMM-REFS" name="DIAG-COMM-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="MULTIPLE-ECU-JOB-REFS" name="MULTIPLE-ECU-JOB-
REFS" minOccurs="0" maxOccurs="1"/>
        </xsd:choice>
        <xsd:attribute use="required" type="xsd:string" name="SEMANTIC"/>
    </xsd:complexType>
    <xsd:complexType name="EXECUTABLES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="EXECUTABLE"
name="EXECUTABLE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="EXPECTED-IDENT">
        <!--Class: EXPECTED-IDENT-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="IDENT-VALUES" name="IDENT-VALUES" minOccurs="1"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="EXPECTED-IDENTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="EXPECTED-
IDENT" name="EXPECTED-IDENT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="EXTERNAL-ACCESS-METHOD">
        <!--Class: EXTERNAL-ACCESS-METHOD-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="xsd:ID" name="ID"/>
                <xsd:attribute use="optional" type="xsd:string" name="OID"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="EXTERNAL-DOC">
        <!--Class: EXTERNAL-DOC-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="xsd:anyURI" name="HREF"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="EXTERNAL-DOCS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="EXTERNAL-DOC"
name="EXTERNAL-DOC"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="EXTERN-FLASHDATA">
        <!--Class: EXTERN-FLASHDATA-->
        <xsd:complexContent>
            <xsd:extension base="FLASHDATA">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="DATAFILE"
name="DATAFILE"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="FIELD">
```

```
        <!--Class: FIELD-->
        <xsd:complexContent>
            <xsd:extension base="COMPLEX-DOP">
                <xsd:choice>
                    <xsd:element maxOccurs="1" minOccurs="1" name="BASIC-
STRUCTURE-REF" type="ODXLINK"/>
                    <xsd:element maxOccurs="1" minOccurs="1" name="ENV-DATA-DESC-
REF" type="ODXLINK"/>
                </xsd:choice>
                <xsd:attribute default="true" use="optional" type="xsd:boolean"
name="IS-VISIBLE"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="FILTER">
        <!--Class: FILTER-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="FILTER-START"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="FILTERS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="FILTER"
name="FILTER"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FLASH">
        <!--Class: FLASH-->
        <xsd:complexContent>
            <xsd:extension base="ODX-CATEGORY">
                <xsd:sequence>
                    <xsd:element type="ECU-MEMS" name="ECU-MEMS" minOccurs="0"
maxOccurs="1"/>
                    <xsd:element type="ECU-MEM-CONNECTORS" name="ECU-MEM-
CONNECTORS" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FLASH-CLASS">
        <!--Class: FLASH-CLASS-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="FLASH-CLASS-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="FLASH-CLASS-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="FLASH-CLASSS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="FLASH-CLASS"
name="FLASH-CLASS"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="FLASHDATA">
        <!--Class: FLASHDATA-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
```

```
            <xsd:element maxOccurs="1" minOccurs="1" type="DATAFORMAT"
name="DATAFORMAT"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ENCRYPT-COMPRESS-
METHOD" name="ENCRYPT-COMPRESS-METHOD"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="FLASHDATAS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="FLASHDATA"
name="FLASHDATA"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FUNCT-CLASS">
        <!--Class: FUNCT-CLASS-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="FUNCT-CLASS-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="FUNCT-CLASS-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="FUNCT-CLASSS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="FUNCT-CLASS"
name="FUNCT-CLASS"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FUNCTIONAL-GROUP">
        <!--Class: FUNCTIONAL-GROUP-->
        <xsd:complexContent>
            <xsd:extension base="HIERARCHY-ELEMENT">
                <xsd:sequence>
                    <xsd:element type="DIAG-VARIABLES" name="DIAG-VARIABLES"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="VARIABLE-GROUPS" name="VARIABLE-GROUPS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="PARENT-REFS"
name="PARENT-REFS"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FUNCTIONAL-GROUP-REF">
        <!--Class: FUNCTIONAL-GROUP-REF-->
        <xsd:complexContent>
            <xsd:extension base="PARENT-REF"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="FUNCTIONAL-GROUPS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="FUNCTIONAL-
GROUP" name="FUNCTIONAL-GROUP"/>
        </xsd:sequence>
    </xsd:complexType>
```

```
    <xsd:complexType abstract="false" name="FUNCTION-DICTIONARY">
        <!--Class: FUNCTION-DICTIONARY-->
        <xsd:complexContent>
            <xsd:extension base="ODX-CATEGORY">
                <xsd:sequence>
                    <xsd:element type="FUNCTION-NODES" name="FUNCTION-NODES"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="FUNCTION-NODE-GROUPS" name="FUNCTION-NODE-
GROUPS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="ADDITIONAL-AUDIENCES" name="ADDITIONAL-
AUDIENCES" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FUNCTION-IN-PARAM">
        <!--Class: FUNCTION-IN-PARAM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="UNIT-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="PHYSICAL-TYPE"
name="PHYSICAL-TYPE"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="IN-PARAM-IF-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="FUNCTION-IN-PARAMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="FUNCTION-IN-
PARAM" name="FUNCTION-IN-PARAM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FUNCTION-NODE">
        <!--Class: FUNCTION-NODE-->
        <xsd:complexContent>
            <xsd:extension base="BASE-FUNCTION-NODE">
                <xsd:sequence>
                    <xsd:element type="FUNCTION-NODES" name="FUNCTION-NODES"
minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FUNCTION-NODE-GROUP">
        <!--Class: FUNCTION-NODE-GROUP-->
        <xsd:complexContent>
            <xsd:extension base="BASE-FUNCTION-NODE">
                <xsd:sequence>
                    <xsd:element type="FUNCTION-NODE-REFS" name="FUNCTION-NODE-
REFS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="FUNCTION-NODE-GROUPS" name="FUNCTION-NODE-
GROUPS" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="FUNCTION-NODE-GROUPS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="FUNCTION-NODE-
GROUP" name="FUNCTION-NODE-GROUP"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="FUNCTION-NODE-REFS">
        <!-- Automatically generated wrapper element type -->
```

```
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="FUNCTION-NODE-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="FUNCTION-NODES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="FUNCTION-NODE"
name="FUNCTION-NODE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FUNCTION-OUT-PARAM">
        <!--Class: FUNCTION-OUT-PARAM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="UNIT-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="PHYSICAL-TYPE"
name="PHYSICAL-TYPE"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="OUT-PARAM-IF-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="FUNCTION-OUT-PARAMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="FUNCTION-OUT-
PARAM" name="FUNCTION-OUT-PARAM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FW-CHECKSUM">
        <!--Class: FW-CHECKSUM-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="SESSION-SUB-ELEM-TYPE"
name="TYPE"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="FW-SIGNATURE">
        <!--Class: FW-SIGNATURE-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="SESSION-SUB-ELEM-TYPE"
name="TYPE"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="GATEWAY-LOGICAL-LINK">
        <!--Class: GATEWAY-LOGICAL-LINK-->
        <xsd:complexContent>
            <xsd:extension base="LOGICAL-LINK">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="SEMANTIC"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="GATEWAY-LOGICAL-LINK-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="GATEWAY-
LOGICAL-LINK-REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
```

```
    <xsd:complexType abstract="false" name="GLOBAL-NEG-RESPONSE">
        <!--Class: GLOBAL-NEG-RESPONSE-->
        <xsd:complexContent>
            <xsd:extension base="RESPONSE"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="GLOBAL-NEG-RESPONSES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="GLOBAL-NEG-
RESPONSE" name="GLOBAL-NEG-RESPONSE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="GROUP-MEMBER">
        <!--Class: GROUP-MEMBER-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="BASE-VARIANT-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="FUNCT-RESOLUTION-LINK-
REF" type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="PHYS-RESOLUTION-LINK-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="GROUP-MEMBERS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="GROUP-MEMBER"
name="GROUP-MEMBER"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="HIERARCHY-ELEMENT">
        <!--Class: HIERARCHY-ELEMENT-->
        <xsd:complexContent>
            <xsd:extension base="DIAG-LAYER">
                <xsd:sequence>
                    <xsd:element type="COMPARAM-REFS" name="COMPARAM-REFS"
minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="IDENT-DESC">
        <!--Class: IDENT-DESC-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="DIAG-COMM-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="IDENT-IF-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="OUT-PARAM-IF-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="IDENT-DESCS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="IDENT-DESC"
name="IDENT-DESC"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="IDENT-VALUE">
        <!--Class: IDENT-VALUE-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="IDENT-VALUE-TYPE"
name="TYPE"/>
            </xsd:extension>
```

```
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="IDENT-VALUES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="IDENT-VALUE"
name="IDENT-VALUE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="IDENT-VALUE-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="A_UINT32"/>
            <xsd:enumeration value="A_BYTEFIELD"/>
            <xsd:enumeration value="A_ASCIISTRING"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="IMPORT-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="IMPORT-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="INFO-COMPONENT">
        <!--Class: INFO-COMPONENT-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="MATCHING-COMPONENTS" name="MATCHING-COMPONENTS"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="INFO-COMPONENT-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="INFO-
COMPONENT-REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="INFO-COMPONENTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="INFO-
COMPONENT" name="INFO-COMPONENT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="IN-PARAM-IF-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="IN-PARAM-IF-
SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="INPUT-PARAM">
        <!--Class: INPUT-PARAM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="PHYSICAL-DEFAULT-VALUE"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="DOP-BASE-REF"
type="ODXLINK"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="optional" type="xsd:string" name="SEMANTIC"/>
    </xsd:complexType>
    <xsd:complexType name="INPUT-PARAMS">
```

```
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="INPUT-PARAM"
name="INPUT-PARAM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="INTERNAL-CONSTR">
        <!--Class: INTERNAL-CONSTR-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="LIMIT" name="LOWER-
LIMIT"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="LIMIT" name="UPPER-
LIMIT"/>
            <xsd:element type="SCALE-CONSTRS" name="SCALE-CONSTRS" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="INTERN-FLASHDATA">
        <!--Class: INTERN-FLASHDATA-->
        <xsd:complexContent>
            <xsd:extension base="FLASHDATA">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="DATA"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:simpleType name="INTERVAL-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="OPEN"/>
            <xsd:enumeration value="CLOSED"/>
            <xsd:enumeration value="INFINITE"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="ITEM-VALUE">
        <!--Class: ITEM-VALUE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="PHYS-CONSTANT-VALUE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="TEXT" name="MEANING"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="KEY"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="RULE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="TEXT"
name="DESCRIPTION"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="AUDIENCE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ITEM-VALUES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="ITEM-VALUE"
name="ITEM-VALUE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="LAYER-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="LAYER-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="LEADING-LENGTH-INFO-TYPE">
```

```
        <!--Class: LEADING-LENGTH-INFO-TYPE-->
        <xsd:complexContent>
            <xsd:extension base="DIAG-CODED-TYPE">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="BIT-LENGTH">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:unsignedInt">
                                <xsd:minExclusive value="0"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="LENGTH-KEY">
        <!--Class: LENGTH-KEY-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
                <xsd:choice maxOccurs="1" minOccurs="1">
                    <xsd:element name="DOP-REF" type="ODXLINK"/>
                    <xsd:element name="DOP-SNREF" type="SNREF"/>
                </xsd:choice>
                <xsd:attribute use="required" type="xsd:ID" name="ID"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="LIMIT">
        <!--Class: LIMIT-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute default="CLOSED" use="optional" type="INTERVAL-
TYPE" name="INTERVAL-TYPE"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="LINK-COMPARAM-REF">
        <!--Class: LINK-COMPARAM-REF-->
        <xsd:sequence>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" type="SIMPLE-VALUE"
name="SIMPLE-VALUE"/>
                <xsd:element maxOccurs="1" minOccurs="1" type="COMPLEX-VALUE"
name="COMPLEX-VALUE"/>
            </xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION"
name="DESC"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="ODXLINK-ATTR"/>
    </xsd:complexType>
    <xsd:complexType name="LINK-COMPARAM-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="LINK-COMPARAM-
REF" name="LINK-COMPARAM-REF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="LINKED-DTC-DOP">
        <!--Class: LINKED-DTC-DOP-->
        <xsd:sequence>
            <xsd:element type="NOT-INHERITED-DTC-SNREFS" name="NOT-INHERITED-DTC-
SNREFS" minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="DTC-DOP-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="LINKED-DTC-DOPS">
```

```
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="LINKED-DTC-
DOP" name="LINKED-DTC-DOP"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="LOGICAL-LINK">
        <!--Class: LOGICAL-LINK-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="GATEWAY-LOGICAL-LINK-REFS" name="GATEWAY-LOGICAL-
LINK-REFS" minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="PHYSICAL-VEHICLE-LINK-
REF" type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="PROTOCOL-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="FUNCTIONAL-GROUP-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="BASE-VARIANT-REF"
type="ODXLINK"/>
            <xsd:element type="ECU-PROXY-REFS" name="ECU-PROXY-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="LINK-COMPARAM-REFS" name="LINK-COMPARAM-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="PROT-STACK-SNREF"
type="SNREF"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="LOGICAL-LINKS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="LOGICAL-LINK"
name="LOGICAL-LINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="LONG-NAME">
        <!-- Automatically generated wrapper element type -->
        <xsd:simpleContent>
            <xsd:restriction base="TEXT">
                <xsd:maxLength value="255"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MATCHING-BASE-VARIANT-PARAMETER">
        <!--Class: MATCHING-BASE-VARIANT-PARAMETER-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="EXPECTED-VALUE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:boolean"
name="USE-PHYSICAL-ADDRESSING"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="DIAG-COMM-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="OUT-PARAM-IF-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="MATCHING-BASE-VARIANT-PARAMETERS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="MATCHING-BASE-
VARIANT-PARAMETER" name="MATCHING-BASE-VARIANT-PARAMETER"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MATCHING-COMPONENT">
        <!--Class: MATCHING-COMPONENT-->
```

```
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="EXPECTED-VALUE"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="OUT-PARAM-IF-SNREF"
type="SNREF"/>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" name="MULTIPLE-ECU-JOB-
REF" type="ODXLINK"/>
                <xsd:element maxOccurs="1" minOccurs="1" name="DIAG-COMM-REF"
type="ODXLINK"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="MATCHING-COMPONENTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="MATCHING-
COMPONENT" name="MATCHING-COMPONENT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MATCHING-PARAMETER">
        <!--Class: MATCHING-PARAMETER-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="EXPECTED-VALUE"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="DIAG-COMM-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="OUT-PARAM-IF-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="MATCHING-PARAMETERS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="MATCHING-
PARAMETER" name="MATCHING-PARAMETER"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MATCHING-REQUEST-PARAM">
        <!--Class: MATCHING-REQUEST-PARAM-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:int"
name="REQUEST-BYTE-POS"/>
                    <xsd:element maxOccurs="1" minOccurs="1"
type="xsd:unsignedInt" name="BYTE-LENGTH"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MEM">
        <!--Class: MEM-->
        <xsd:sequence>
            <xsd:element type="SESSIONS" name="SESSIONS" minOccurs="1"
maxOccurs="1"/>
            <xsd:element type="DATABLOCKS" name="DATABLOCKS" minOccurs="1"
maxOccurs="1"/>
            <xsd:element type="FLASHDATAS" name="FLASHDATAS" minOccurs="1"
maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MEMBER-LOGICAL-LINK">
        <!--Class: MEMBER-LOGICAL-LINK-->
        <xsd:complexContent>
            <xsd:extension base="LOGICAL-LINK"/>
        </xsd:complexContent>
```

```
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MIN-MAX-LENGTH-TYPE">
        <!--Class: MIN-MAX-LENGTH-TYPE-->
        <xsd:complexContent>
            <xsd:extension base="DIAG-CODED-TYPE">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="0"
type="xsd:unsignedInt" name="MAX-LENGTH"/>
                    <xsd:element maxOccurs="1" minOccurs="1"
type="xsd:unsignedInt" name="MIN-LENGTH"/>
                </xsd:sequence>
                <xsd:attribute use="required" type="TERMINATION"
name="TERMINATION"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MODEL-YEAR">
        <!--Class: MODEL-YEAR-->
        <xsd:complexContent>
            <xsd:extension base="INFO-COMPONENT"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MODIFICATION">
        <!--Class: MODIFICATION-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="CHANGE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="REASON"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="MODIFICATIONS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="MODIFICATION"
name="MODIFICATION"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MULTIPLE-ECU-JOB">
        <!--Class: MULTIPLE-ECU-JOB-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="FUNCT-CLASS-REFS" name="FUNCT-CLASS-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="PROG-CODES" name="PROG-CODES" minOccurs="1"
maxOccurs="1"/>
            <xsd:element type="INPUT-PARAMS" name="INPUT-PARAMS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="OUTPUT-PARAMS" name="OUTPUT-PARAMS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="NEG-OUTPUT-PARAMS" name="NEG-OUTPUT-PARAMS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="DIAG-LAYER-REFS" name="DIAG-LAYER-REFS"
minOccurs="1" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="AUDIENCE"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="optional" type="xsd:string" name="SEMANTIC"/>
        <xsd:attribute default="true" use="optional" type="xsd:boolean" name="IS-
EXECUTABLE"/>
        <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-REDUCED-RESULT-ENABLED"/>
```

```
    </xsd:complexType>
    <xsd:complexType name="MULTIPLE-ECU-JOB-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="MULTIPLE-ECU-
JOB-REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="MULTIPLE-ECU-JOBS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="MULTIPLE-ECU-
JOB" name="MULTIPLE-ECU-JOB"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MULTIPLE-ECU-JOB-SPEC">
        <!--Class: MULTIPLE-ECU-JOB-SPEC-->
        <xsd:complexContent>
            <xsd:extension base="ODX-CATEGORY">
                <xsd:sequence>
                    <xsd:element type="MULTIPLE-ECU-JOBS" name="MULTIPLE-ECU-
JOBS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="DIAG-DATA-
DICTIONARY-SPEC" name="DIAG-DATA-DICTIONARY-SPEC"/>
                    <xsd:element type="FUNCT-CLASSS" name="FUNCT-CLASSS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="ADDITIONAL-AUDIENCES" name="ADDITIONAL-
AUDIENCES" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="MUX">
        <!--Class: MUX-->
        <xsd:complexContent>
            <xsd:extension base="COMPLEX-DOP">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1"
type="xsd:unsignedInt" name="BYTE-POSITION"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="SWITCH-KEY"
name="SWITCH-KEY"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="DEFAULT-CASE"
name="DEFAULT-CASE"/>
                    <xsd:element type="CASES" name="CASES" minOccurs="0"
maxOccurs="1"/>
                </xsd:sequence>
                <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-VISIBLE"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="MUXS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="MUX"
name="MUX"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="NEG-OFFSET">
        <!--Class: NEG-OFFSET-->
        <xsd:complexContent>
            <xsd:extension base="TARGET-ADDR-OFFSET">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="NEGATIVE-OFFSET"/>
                </xsd:sequence>
            </xsd:extension>
```

```
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="NEG-OUTPUT-PARAM">
        <!--Class: NEG-OUTPUT-PARAM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="DOP-BASE-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="NEG-OUTPUT-PARAMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="NEG-OUTPUT-
PARAM" name="NEG-OUTPUT-PARAM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="NEG-RESPONSE">
        <!--Class: NEG-RESPONSE-->
        <xsd:complexContent>
            <xsd:extension base="RESPONSE"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="NEG-RESPONSE-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="NEG-RESPONSE-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="NEG-RESPONSES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="NEG-RESPONSE"
name="NEG-RESPONSE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="NOT-INHERITED-DIAG-COMM">
        <!--Class: NOT-INHERITED-DIAG-COMM-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="DIAG-COMM-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="NOT-INHERITED-DIAG-COMMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="NOT-INHERITED-
DIAG-COMM" name="NOT-INHERITED-DIAG-COMM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="NOT-INHERITED-DOP">
        <!--Class: NOT-INHERITED-DOP-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="DOP-BASE-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="NOT-INHERITED-DOPS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="NOT-INHERITED-
DOP" name="NOT-INHERITED-DOP"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="NOT-INHERITED-DTC-SNREFS">
        <!-- Automatically generated wrapper element type -->
```

```
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="NOT-INHERITED-
DTC-SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="NOT-INHERITED-TABLE">
        <!--Class: NOT-INHERITED-TABLE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="NOT-INHERITED-TABLES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="NOT-INHERITED-
TABLE" name="NOT-INHERITED-TABLE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="NOT-INHERITED-VARIABLE">
        <!--Class: NOT-INHERITED-VARIABLE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="DIAG-VARIABLE-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="NOT-INHERITED-VARIABLES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="NOT-INHERITED-
VARIABLE" name="NOT-INHERITED-VARIABLE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="NRC-CONST">
        <!--Class: NRC-CONST-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
                <xsd:sequence>
                    <xsd:element type="CODED-VALUES" name="CODED-VALUES"
minOccurs="1" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="1" type="DIAG-CODED-
TYPE" name="DIAG-CODED-TYPE"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="ODX">
        <!--Class: ODX-->
        <xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="1" type="DIAG-LAYER-CONTAINER"
name="DIAG-LAYER-CONTAINER"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="COMPARAM-SPEC"
name="COMPARAM-SPEC"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="VEHICLE-INFO-SPEC"
name="VEHICLE-INFO-SPEC"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="FLASH" name="FLASH"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="ECU-CONFIG" name="ECU-
CONFIG"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="MULTIPLE-ECU-JOB-SPEC"
name="MULTIPLE-ECU-JOB-SPEC"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="COMPARAM-SUBSET"
name="COMPARAM-SUBSET"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="FUNCTION-DICTIONARY"
name="FUNCTION-DICTIONARY"/>
        </xsd:choice>
        <xsd:attribute fixed="2.1.0" use="required" type="xsd:string"
name="MODEL-VERSION"/>
```

```
    </xsd:complexType>
    <xsd:complexType abstract="true" name="ODX-CATEGORY">
        <!--Class: ODX-CATEGORY-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element type="COMPANY-DATAS" name="COMPANY-DATAS" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="ODXLINK">
        <xsd:attributeGroup ref="ODXLINK-ATTR"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="OEM">
        <!--Class: OEM-->
        <xsd:complexContent>
            <xsd:extension base="INFO-COMPONENT"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="OPTION-ITEM">
        <!--Class: OPTION-ITEM-->
        <xsd:complexContent>
            <xsd:extension base="CONFIG-ITEM">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="PHYSICAL-DEFAULT-VALUE"/>
                    <xsd:element type="ITEM-VALUES" name="ITEM-VALUES"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="WRITE-AUDIENCE"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="READ-AUDIENCE"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="OPTION-ITEMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="OPTION-ITEM"
name="OPTION-ITEM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="OUT-PARAM-IF-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="OUT-PARAM-IF-
SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="OUTPUT-PARAM">
        <!--Class: OUTPUT-PARAM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="DOP-BASE-REF"
type="ODXLINK"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="optional" type="xsd:string" name="SEMANTIC"/>
    </xsd:complexType>
    <xsd:complexType name="OUTPUT-PARAMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
```

```
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="OUTPUT-PARAM"
name="OUTPUT-PARAM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="OWN-IDENT">
        <!--Class: OWN-IDENT-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="IDENT-VALUE"
name="IDENT-VALUE"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="OWN-IDENTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="OWN-IDENT"
name="OWN-IDENT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="PARAM">
        <!--Class: PARAM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="optional" type="xsd:string" name="SEMANTIC"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="PARAM-LENGTH-INFO-TYPE">
        <!--Class: PARAM-LENGTH-INFO-TYPE-->
        <xsd:complexContent>
            <xsd:extension base="DIAG-CODED-TYPE">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="LENGTH-KEY-
REF" type="ODXLINK"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="PARAMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="PARAM"
name="PARAM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="PARENT-REF">
        <!--Class: PARENT-REF-->
        <xsd:sequence>
            <xsd:element type="NOT-INHERITED-DIAG-COMMS" name="NOT-INHERITED-
DIAG-COMMS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="NOT-INHERITED-VARIABLES" name="NOT-INHERITED-
VARIABLES" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="NOT-INHERITED-DOPS" name="NOT-INHERITED-DOPS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="NOT-INHERITED-TABLES" name="NOT-INHERITED-TABLES"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="ODXLINK-ATTR"/>
    </xsd:complexType>
    <xsd:complexType name="PARENT-REFS">
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="PARENT-REF"
name="PARENT-REF"/>
        </xsd:sequence>
    </xsd:complexType>
```

```
    <xsd:complexType abstract="false" name="PHYS-CONST">
        <!--Class: PHYS-CONST-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="PHYS-CONSTANT-VALUE"/>
                    <xsd:choice maxOccurs="1" minOccurs="1">
                        <xsd:element name="DOP-REF" type="ODXLINK"/>
                        <xsd:element name="DOP-SNREF" type="SNREF"/>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:simpleType name="PHYSICAL-DATA-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="A_INT32"/>
            <xsd:enumeration value="A_UINT32"/>
            <xsd:enumeration value="A_FLOAT32"/>
            <xsd:enumeration value="A_FLOAT64"/>
            <xsd:enumeration value="A_UNICODE2STRING"/>
            <xsd:enumeration value="A_BYTEFIELD"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="PHYSICAL-DIMENSION">
        <!--Class: PHYSICAL-DIMENSION-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:int" name="LENGTH-
EXP"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:int" name="MASS-
EXP"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:int" name="TIME-
EXP"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:int"
name="CURRENT-EXP"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:int"
name="TEMPERATURE-EXP"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:int" name="MOLAR-
AMOUNT-EXP"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:int"
name="LUMINOUS-INTENSITY-EXP"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="PHYSICAL-DIMENSIONS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="PHYSICAL-
DIMENSION" name="PHYSICAL-DIMENSION"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="PHYSICAL-TYPE">
        <!--Class: PHYSICAL-TYPE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:unsignedInt"
name="PRECISION"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="PHYSICAL-DATA-TYPE" name="BASE-DATA-
TYPE"/>
        <xsd:attribute use="optional" type="RADIX" name="DISPLAY-RADIX"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="PHYSICAL-VEHICLE-LINK">
        <!--Class: PHYSICAL-VEHICLE-LINK-->
        <xsd:sequence>
```

```
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="VEHICLE-CONNECTOR-PIN-REFS" name="VEHICLE-
CONNECTOR-PIN-REFS" minOccurs="1" maxOccurs="1"/>
            <xsd:element type="LINK-COMPARAM-REFS" name="LINK-COMPARAM-REFS"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="required" type="xsd:string" name="TYPE"/>
    </xsd:complexType>
    <xsd:complexType name="PHYSICAL-VEHICLE-LINKS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="PHYSICAL-
VEHICLE-LINK" name="PHYSICAL-VEHICLE-LINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="PHYS-MEM">
        <!--Class: PHYS-MEM-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="PHYS-SEGMENTS" name="PHYS-SEGMENTS" minOccurs="1"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="PHYS-SEGMENT">
        <!--Class: PHYS-SEGMENT-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:hexBinary"
name="FILLBYTE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:unsignedInt"
name="BLOCK-SIZE"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="START-ADDRESS"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="PHYS-SEGMENTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="PHYS-SEGMENT"
name="PHYS-SEGMENT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="PIN-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="HI"/>
            <xsd:enumeration value="LOW"/>
            <xsd:enumeration value="K"/>
            <xsd:enumeration value="L"/>
            <xsd:enumeration value="TX"/>
            <xsd:enumeration value="RX"/>
            <xsd:enumeration value="PLUS"/>
            <xsd:enumeration value="MINUS"/>
            <xsd:enumeration value="SINGLE"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="true" name="POSITIONABLE-PARAM">
        <!--Class: POSITIONABLE-PARAM-->
        <xsd:complexContent>
            <xsd:extension base="PARAM">
                <xsd:sequence>
```

```
                    <xsd:element maxOccurs="1" minOccurs="0"
type="xsd:unsignedInt" name="BYTE-POSITION"/>
                    <xsd:element maxOccurs="1" minOccurs="0" name="BIT-POSITION">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:unsignedInt">
                                <xsd:maxExclusive value="8"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="POS-OFFSET">
        <!--Class: POS-OFFSET-->
        <xsd:complexContent>
            <xsd:extension base="TARGET-ADDR-OFFSET">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="POSITIVE-OFFSET"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="POS-RESPONSE">
        <!--Class: POS-RESPONSE-->
        <xsd:complexContent>
            <xsd:extension base="RESPONSE"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="POS-RESPONSE-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="POS-RESPONSE-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="POS-RESPONSES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="POS-RESPONSE"
name="POS-RESPONSE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="POS-RESPONSE-SUPPRESSABLE">
        <!--Class: POS-RESPONSE-SUPPRESSABLE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="BIT-MASK"/>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" name="CODED-CONST-SNREF"
type="SNREF"/>
                <xsd:element maxOccurs="1" minOccurs="1" name="VALUE-SNREF"
type="SNREF"/>
                <xsd:element maxOccurs="1" minOccurs="1" name="PHYS-CONST-SNREF"
type="SNREF"/>
                <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-KEY-SNREF"
type="SNREF"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="POSSIBLE-PHYSICAL-LINK-TYPES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" type="xsd:string"
name="POSSIBLE-PHYSICAL-LINK-TYPE"/>
        </xsd:sequence>
```

```
    </xsd:complexType>
    <xsd:complexType abstract="false" name="PRE-CONDITION-STATE-REF">
        <!--Class: PRE-CONDITION-STATE-REF-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="VALUE"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="IN-PARAM-IF-SNREF"
type="SNREF"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="ODXLINK-ATTR"/>
    </xsd:complexType>
    <xsd:complexType name="PRE-CONDITION-STATE-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="PRE-CONDITION-
STATE-REF" name="PRE-CONDITION-STATE-REF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="PROG-CODE">
        <!--Class: PROG-CODE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="CODE-FILE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="ENCRYPTION"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="SYNTAX"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="REVISION"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="ENTRYPOINT"/>
        </xsd:sequence>
        <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="LATEBOUND-CODE-FILE"/>
    </xsd:complexType>
    <xsd:complexType name="PROG-CODES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="PROG-CODE"
name="PROG-CODE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="PROTOCOL">
        <!--Class: PROTOCOL-->
        <xsd:complexContent>
            <xsd:extension base="HIERARCHY-ELEMENT">
                <xsd:sequence>
                    <xsd:element type="POSSIBLE-PHYSICAL-LINK-TYPES"
name="POSSIBLE-PHYSICAL-LINK-TYPES" minOccurs="0" maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="1" name="COMPARAM-SPEC-
REF" type="ODXLINK"/>
                    <xsd:element maxOccurs="1" minOccurs="0" name="PROT-STACK-
SNREF" type="SNREF"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="PARENT-REFS"
name="PARENT-REFS"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="PROTOCOL-REF">
        <!--Class: PROTOCOL-REF-->
        <xsd:complexContent>
            <xsd:extension base="PARENT-REF"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="PROTOCOLS">
        <!-- Automatically generated wrapper element type -->
```

```
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="PROTOCOL"
name="PROTOCOL"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="PROTOCOL-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="PROTOCOL-
SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="PROT-STACK">
        <!--Class: PROT-STACK-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string" name="PDU-
PROTOCOL-TYPE"/>
            <xsd:element type="COMPARAM-SUBSET-REFS" name="COMPARAM-SUBSET-REFS"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="PROT-STACKS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="PROT-STACK"
name="PROT-STACK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="RADIX">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="HEX"/>
            <xsd:enumeration value="DEC"/>
            <xsd:enumeration value="BIN"/>
            <xsd:enumeration value="OCT"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="READ-DIAG-COMM-CONNECTOR">
        <!--Class: READ-DIAG-COMM-CONNECTOR-->
        <xsd:sequence>
            <xsd:element type="READ-PARAM-VALUES" name="READ-PARAM-VALUES"
minOccurs="0" maxOccurs="1"/>
            <xsd:choice maxOccurs="1" minOccurs="1">
                <xsd:element name="READ-DIAG-COMM-REF" type="ODXLINK"/>
                <xsd:element name="READ-DIAG-COMM-SNREF" type="SNREF"/>
            </xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="1" name="READ-DATA-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="READ-PARAM-VALUE">
        <!--Class: READ-PARAM-VALUE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="PHYS-CONSTANT-VALUE"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="IN-PARAM-IF-SNREF"
type="SNREF"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:string" name="SEMANTIC"/>
    </xsd:complexType>
    <xsd:complexType name="READ-PARAM-VALUES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="READ-PARAM-
VALUE" name="READ-PARAM-VALUE"/>
```

```
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="RELATED-DIAG-COMM-REF">
        <!--Class: RELATED-DIAG-COMM-REF-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="RELATION-TYPE"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="ODXLINK-ATTR"/>
    </xsd:complexType>
    <xsd:complexType name="RELATED-DIAG-COMM-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="RELATED-DIAG-
COMM-REF" name="RELATED-DIAG-COMM-REF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="RELATED-DOC">
        <!--Class: RELATED-DOC-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="XDOC" name="XDOC"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION"
name="DESC"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="RELATED-DOCS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="RELATED-DOC"
name="RELATED-DOC"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="REQUEST">
        <!--Class: REQUEST-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element type="PARAMS" name="PARAMS" minOccurs="1"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="REQUESTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="REQUEST"
name="REQUEST"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="RESERVED">
        <!--Class: RESERVED-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="BIT-LENGTH">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:unsignedInt">
                                <xsd:minExclusive value="0"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="CODED-VALUE"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="DIAG-CODED-
TYPE" name="DIAG-CODED-TYPE"/>
```

```
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="RESPONSE">
        <!--Class: RESPONSE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element type="PARAMS" name="PARAMS" minOccurs="1"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="ROLES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" type="xsd:string"
name="ROLE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="ROW-FRAGMENT">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="KEY"/>
            <xsd:enumeration value="STRUCT"/>
            <xsd:enumeration value="KEY-AND-STRUCT"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="SCALE-CONSTR">
        <!--Class: SCALE-CONSTR-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="TEXT" name="SHORT-
LABEL"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION"
name="DESC"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="LIMIT" name="LOWER-
LIMIT"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="LIMIT" name="UPPER-
LIMIT"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="VALID-TYPE" name="VALIDITY"/>
    </xsd:complexType>
    <xsd:complexType name="SCALE-CONSTRS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SCALE-CONSTR"
name="SCALE-CONSTR"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SD">
        <!--Class: SD-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="optional" type="xsd:string" name="TI"/>
                <xsd:attribute use="optional" type="xsd:string" name="SI"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SDG">
        <!--Class: SDG-->
        <xsd:complexContent>
            <xsd:extension base="SPECIAL-DATA">
                <xsd:sequence>
                    <xsd:choice>
```

```
                        <xsd:element maxOccurs="1" minOccurs="0" name="SDG-
CAPTION-REF" type="ODXLINK"/>
                        <xsd:element maxOccurs="1" minOccurs="0" type="SDG-
CAPTION" name="SDG-CAPTION"/>
                    </xsd:choice>
                    <xsd:choice maxOccurs="unbounded" minOccurs="0">
                        <xsd:element maxOccurs="1" minOccurs="1" type="SDG"
name="SDG"/>
                        <xsd:element maxOccurs="1" minOccurs="1" type="SD"
name="SD"/>
                    </xsd:choice>
                </xsd:sequence>
                <xsd:attribute use="optional" type="xsd:string" name="SI"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SDG-CAPTION">
        <!--Class: SDG-CAPTION-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="SDGS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SDG"
name="SDG"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SECURITY">
        <!--Class: SECURITY-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="SECURITY-METHOD"
name="SECURITY-METHOD"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="FW-SIGNATURE"
name="FW-SIGNATURE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="FW-CHECKSUM" name="FW-
CHECKSUM"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="VALIDITY-FOR"
name="VALIDITY-FOR"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SECURITY-METHOD">
        <!--Class: SECURITY-METHOD-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="SESSION-SUB-ELEM-TYPE"
name="TYPE"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="SECURITYS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SECURITY"
name="SECURITY"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SEGMENT">
        <!--Class: SEGMENT-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:hexBinary"
name="SOURCE-START-ADDRESS"/>
```

```
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:unsignedInt"
name="COMPRESSED-SIZE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="COMPRESSION-METHOD"/>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" type="UNCOMPRESSED-SIZE"
name="UNCOMPRESSED-SIZE"/>
                <xsd:element maxOccurs="1" minOccurs="1" type="SOURCE-END-
ADDRESS" name="SOURCE-END-ADDRESS"/>
            </xsd:choice>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="SEGMENTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SEGMENT"
name="SEGMENT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="SELECTION-TABLE-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:choice maxOccurs="unbounded" minOccurs="1">
            <xsd:element name="SELECTION-TABLE-REF" type="ODXLINK"/>
            <xsd:element name="SELECTION-TABLE-SNREF" type="SNREF"/>
        </xsd:choice>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SESSION">
        <!--Class: SESSION-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="EXPECTED-IDENTS" name="EXPECTED-IDENTS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="CHECKSUMS" name="CHECKSUMS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="SECURITYS" name="SECURITYS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="DATABLOCK-REFS" name="DATABLOCK-REFS"
minOccurs="1" maxOccurs="1"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SESSION-DESC">
        <!--Class: SESSION-DESC-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="PARTNUMBER"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:unsignedInt"
name="PRIORITY"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="SESSION-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="DIAG-COMM-SNREF"
type="SNREF"/>
            <xsd:element type="FLASH-CLASS-REFS" name="FLASH-CLASS-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="AUDIENCE"
name="AUDIENCE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="OWN-IDENT" name="OWN-
IDENT"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="required" type="DIRECTION" name="DIRECTION"/>
```

```
    </xsd:complexType>
    <xsd:complexType name="SESSION-DESCS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SESSION-DESC"
name="SESSION-DESC"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="SESSIONS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SESSION"
name="SESSION"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="SESSION-SUB-ELEM-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="A_UINT32"/>
            <xsd:enumeration value="A_BYTEFIELD"/>
            <xsd:enumeration value="A_ASCIISTRING"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="SIMPLE-VALUE">
        <!--Class: SIMPLE-VALUE-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SINGLE-ECU-JOB">
        <!--Class: SINGLE-ECU-JOB-->
        <xsd:complexContent>
            <xsd:extension base="DIAG-COMM">
                <xsd:sequence>
                    <xsd:element type="PROG-CODES" name="PROG-CODES"
minOccurs="1" maxOccurs="1"/>
                    <xsd:element type="INPUT-PARAMS" name="INPUT-PARAMS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="OUTPUT-PARAMS" name="OUTPUT-PARAMS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="NEG-OUTPUT-PARAMS" name="NEG-OUTPUT-
PARAMS" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
                <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="IS-REDUCED-RESULT-ENABLED"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SIZEDEF-FILTER">
        <!--Class: SIZEDEF-FILTER-->
        <xsd:complexContent>
            <xsd:extension base="FILTER">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="FILTER-SIZE">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:unsignedInt">
                                <xsd:minExclusive value="0"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SIZEDEF-PHYS-SEGMENT">
        <!--Class: SIZEDEF-PHYS-SEGMENT-->
        <xsd:complexContent>
            <xsd:extension base="PHYS-SEGMENT">
```

```
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="SIZE">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:unsignedInt">
                                <xsd:minExclusive value="0"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="SNREF">
        <xsd:attribute use="required" name="SHORT-NAME">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:pattern value="[a-zA-Z0-9_]{1,128}"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SNREF-TO-TABLEROW">
        <!--Class: SNREF-TO-TABLEROW-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-ROW-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SOURCE-END-ADDRESS">
        <!--Class: SOURCE-END-ADDRESS-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:hexBinary"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="SPECIAL-DATA">
        <!--Class: SPECIAL-DATA-->
    </xsd:complexType>
    <xsd:simpleType name="STANDARDISATION-LEVEL">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="STANDARD"/>
            <xsd:enumeration value="OEM-SPECIFIC"/>
            <xsd:enumeration value="OPTIONAL"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="STANDARD-LENGTH-TYPE">
        <!--Class: STANDARD-LENGTH-TYPE-->
        <xsd:complexContent>
            <xsd:extension base="DIAG-CODED-TYPE">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="BIT-LENGTH">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:unsignedInt">
                                <xsd:minExclusive value="0"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:hexBinary"
name="BIT-MASK"/>
                </xsd:sequence>
                <xsd:attribute default="false" use="optional" type="xsd:boolean"
name="CONDENSED"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="STATE">
```

```
        <!--Class: STATE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="STATE-CHART">
        <!--Class: STATE-CHART-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="SEMANTIC"/>
            <xsd:element type="STATE-TRANSITIONS" name="STATE-TRANSITIONS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="START-STATE-SNREF"
type="SNREF"/>
            <xsd:element type="STATES" name="STATES" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="STATE-CHARTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="STATE-CHART"
name="STATE-CHART"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="STATES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="STATE"
name="STATE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="STATE-TRANSITION">
        <!--Class: STATE-TRANSITION-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="SOURCE-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="TARGET-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="EXTERNAL-ACCESS-
METHOD" name="EXTERNAL-ACCESS-METHOD"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="STATE-TRANSITION-REF">
        <!--Class: STATE-TRANSITION-REF-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="VALUE"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="IN-PARAM-IF-SNREF"
type="SNREF"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="ODXLINK-ATTR"/>
    </xsd:complexType>
    <xsd:complexType name="STATE-TRANSITION-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="STATE-
TRANSITION-REF" name="STATE-TRANSITION-REF"/>
        </xsd:sequence>
```

```
    </xsd:complexType>
    <xsd:complexType name="STATE-TRANSITIONS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="STATE-
TRANSITION" name="STATE-TRANSITION"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="STATIC-FIELD">
        <!--Class: STATIC-FIELD-->
        <xsd:complexContent>
            <xsd:extension base="FIELD">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1"
type="xsd:unsignedInt" name="FIXED-NUMBER-OF-ITEMS"/>
                    <xsd:element maxOccurs="1" minOccurs="1"
type="xsd:unsignedInt" name="ITEM-BYTE-SIZE"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="STATIC-FIELDS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="STATIC-FIELD"
name="STATIC-FIELD"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="STRUCTURE">
        <!--Class: STRUCTURE-->
        <xsd:complexContent>
            <xsd:extension base="BASIC-STRUCTURE">
                <xsd:attribute default="true" use="optional" type="xsd:boolean"
name="IS-VISIBLE"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="STRUCTURES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="STRUCTURE"
name="STRUCTURE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SUB-COMPONENT">
        <!--Class: SUB-COMPONENT-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="DTC-SNREFS" name="DTC-SNREFS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="TABLE-ROW-SNREFS" name="TABLE-ROW-SNREFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="ENV-DATA-SNREFS" name="ENV-DATA-SNREFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="SUB-COMPONENT-PATTERNS" name="SUB-COMPONENT-
PATTERNS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="IN-PARAM-IF-SNREFS" name="IN-PARAM-IF-SNREFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="DOP-BASE-SNREFS" name="DOP-BASE-SNREFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="TABLE-SNREFS" name="TABLE-SNREFS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="OUT-PARAM-IF-SNREFS" name="OUT-PARAM-IF-SNREFS"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
```

```
        <xsd:attribute use="optional" type="xsd:string" name="SEMANTIC"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SUB-COMPONENT-PATTERN">
        <!--Class: SUB-COMPONENT-PATTERN-->
        <xsd:sequence>
            <xsd:element type="MATCHING-PARAMETERS" name="MATCHING-PARAMETERS"
minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="SUB-COMPONENT-PATTERNS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SUB-COMPONENT-
PATTERN" name="SUB-COMPONENT-PATTERN"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="SUB-COMPONENTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SUB-COMPONENT"
name="SUB-COMPONENT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SUPPORTED-DYN-ID">
        <!--Class: SUPPORTED-DYN-ID-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:hexBinary"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="SUPPORTED-DYN-IDS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SUPPORTED-DYN-
ID" name="SUPPORTED-DYN-ID"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SWITCH-KEY">
        <!--Class: SWITCH-KEY-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:unsignedInt"
name="BYTE-POSITION"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="BIT-POSITION">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedInt">
                        <xsd:maxExclusive value="8"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element maxOccurs="1" minOccurs="1" name="DATA-OBJECT-PROP-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SW-VARIABLE">
        <!--Class: SW-VARIABLE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="ORIGIN"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="SW-VARIABLES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SW-VARIABLE"
name="SW-VARIABLE"/>
        </xsd:sequence>
```

```
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SYSTEM">
        <!--Class: SYSTEM-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
                <xsd:choice maxOccurs="1" minOccurs="1">
                    <xsd:element name="DOP-REF" type="ODXLINK"/>
                    <xsd:element name="DOP-SNREF" type="SNREF"/>
                </xsd:choice>
                <xsd:attribute use="required" type="xsd:string" name="SYSPARAM"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="SYSTEM-ITEM">
        <!--Class: SYSTEM-ITEM-->
        <xsd:complexContent>
            <xsd:extension base="CONFIG-ITEM">
                <xsd:attribute use="required" type="xsd:string" name="SYSPARAM"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="SYSTEM-ITEMS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="SYSTEM-ITEM"
name="SYSTEM-ITEM"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="TABLE">
        <!--Class: TABLE-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string" name="KEY-
LABEL"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="STRUCT-LABEL"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="KEY-DOP-REF"
type="ODXLINK"/>
            <xsd:group maxOccurs="unbounded" minOccurs="1" ref="ROW-WRAPPER"/>
            <xsd:element type="TABLE-DIAG-COMM-CONNECTORS" name="TABLE-DIAG-COMM-
CONNECTORS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="optional" type="xsd:string" name="SEMANTIC"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="TABLE-DIAG-COMM-CONNECTOR">
        <!--Class: TABLE-DIAG-COMM-CONNECTOR-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="SEMANTIC"/>
            <xsd:choice maxOccurs="1" minOccurs="1">
                <xsd:element name="DIAG-COMM-REF" type="ODXLINK"/>
                <xsd:element name="DIAG-COMM-SNREF" type="SNREF"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="TABLE-DIAG-COMM-CONNECTORS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="TABLE-DIAG-
COMM-CONNECTOR" name="TABLE-DIAG-COMM-CONNECTOR"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="TABLE-ENTRY">
```

```
        <!--Class: TABLE-ENTRY-->
        <xsd:complexContent>
            <xsd:extension base="PARAM">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" type="ROW-FRAGMENT"
name="TARGET"/>
                    <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-ROW-REF"
type="ODXLINK"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="TABLE-KEY">
        <!--Class: TABLE-KEY-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
                <xsd:choice>
                    <xsd:group maxOccurs="1" minOccurs="1" ref="ODXLINK-TO-
TABLE"/>
                    <xsd:group maxOccurs="1" minOccurs="1" ref="SNREF-TO-TABLE"/>
                    <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-ROW-REF"
type="ODXLINK"/>
                </xsd:choice>
                <xsd:attribute use="required" type="xsd:ID" name="ID"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="TABLE-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="TABLE-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="TABLE-ROW">
        <!--Class: TABLE-ROW-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="KEY"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="STRUCTURE-REF"
type="ODXLINK"/>
            <xsd:element type="SDGS" name="SDGS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="TABLE-ROW-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="TABLE-ROW-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="TABLE-ROW-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="TABLE-ROW-
SNREF" type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="TABLES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="TABLE"
name="TABLE"/>
        </xsd:sequence>
```

```
    </xsd:complexType>
    <xsd:complexType name="TABLE-SNREFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="TABLE-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="TABLE-STRUCT">
        <!--Class: TABLE-STRUCT-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-KEY-REF"
type="ODXLINK"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="TARGET-ADDR-OFFSET">
        <!--Class: TARGET-ADDR-OFFSET-->
    </xsd:complexType>
    <xsd:complexType abstract="false" name="TEAM-MEMBER">
        <!--Class: TEAM-MEMBER-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="ROLES" name="ROLES" minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="DEPARTMENT"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="ADDRESS"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="ZIP"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="CITY"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="PHONE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="FAX"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="EMAIL"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="TEAM-MEMBERS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="TEAM-MEMBER"
name="TEAM-MEMBER"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="TERMINATION">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="END-OF-PDU"/>
            <xsd:enumeration value="ZERO"/>
            <xsd:enumeration value="HEX-FF"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="TEXT">
        <!--Class: TEXT-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="optional" type="xsd:string" name="TI"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
```

```
    <xsd:simpleType name="TRANS-MODE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="SEND-ONLY"/>
            <xsd:enumeration value="RECEIVE-ONLY"/>
            <xsd:enumeration value="SEND-AND-RECEIVE"/>
            <xsd:enumeration value="SEND-OR-RECEIVE"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="UNCOMPRESSED-SIZE">
        <!--Class: UNCOMPRESSED-SIZE-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:unsignedInt"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="true" name="UNION-VALUE">
        <!--Class: UNION-VALUE-->
    </xsd:complexType>
    <xsd:complexType abstract="false" name="UNIT">
        <!--Class: UNIT-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="DISPLAY-NAME"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:double"
name="FACTOR-SI-TO-UNIT"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:double"
name="OFFSET-SI-TO-UNIT"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="PHYSICAL-DIMENSION-
REF" type="ODXLINK"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="UNIT-GROUP">
        <!--Class: UNIT-GROUP-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="UNIT-GROUP-CATEGORY"
name="CATEGORY"/>
            <xsd:element type="UNIT-REFS" name="UNIT-REFS" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:simpleType name="UNIT-GROUP-CATEGORY">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="COUNTRY"/>
            <xsd:enumeration value="EQUIV-UNITS"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="UNIT-GROUPS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="UNIT-GROUP"
name="UNIT-GROUP"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="UNIT-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="UNIT-REF"
type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="UNITS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
```

```
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="UNIT"
name="UNIT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="UNIT-SPEC">
        <!--Class: UNIT-SPEC-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" type="ADMIN-DATA"
name="ADMIN-DATA"/>
            <xsd:element type="UNIT-GROUPS" name="UNIT-GROUPS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="UNITS" name="UNITS" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="PHYSICAL-DIMENSIONS" name="PHYSICAL-DIMENSIONS"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="USAGE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="ECU-SOFTWARE"/>
            <xsd:enumeration value="ECU-COMM"/>
            <xsd:enumeration value="APPLICATION"/>
            <xsd:enumeration value="TESTER"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="V">
        <!--Class: V-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:double"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VALID-BASE-VARIANT">
        <!--Class: VALID-BASE-VARIANT-->
        <xsd:sequence>
            <xsd:element type="ECU-VARIANT-SNREFS" name="ECU-VARIANT-SNREFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="BASE-VARIANT-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="VALID-BASE-VARIANTS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="VALID-BASE-
VARIANT" name="VALID-BASE-VARIANT"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VALIDITY-FOR">
        <!--Class: VALIDITY-FOR-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="required" type="SESSION-SUB-ELEM-TYPE"
name="TYPE"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:simpleType name="VALID-TYPE">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="VALID"/>
            <xsd:enumeration value="NOT-VALID"/>
            <xsd:enumeration value="NOT-DEFINED"/>
            <xsd:enumeration value="NOT-AVAILABLE"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType abstract="false" name="VALUE">
        <!--Class: VALUE-->
        <xsd:complexContent>
            <xsd:extension base="POSITIONABLE-PARAM">
```

```
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="PHYSICAL-DEFAULT-VALUE"/>
                    <xsd:choice maxOccurs="1" minOccurs="1">
                        <xsd:element name="DOP-REF" type="ODXLINK"/>
                        <xsd:element name="DOP-SNREF" type="SNREF"/>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VARIABLE-GROUP">
        <!--Class: VARIABLE-GROUP-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType name="VARIABLE-GROUPS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="VARIABLE-
GROUP" name="VARIABLE-GROUP"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VEHICLE-CONNECTOR">
        <!--Class: VEHICLE-CONNECTOR-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="VEHICLE-CONNECTOR-PINS" name="VEHICLE-CONNECTOR-
PINS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VEHICLE-CONNECTOR-PIN">
        <!--Class: VEHICLE-CONNECTOR-PIN-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:unsignedInt"
name="PIN_NUMBER"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:ID" name="ID"/>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
        <xsd:attribute use="required" type="PIN-TYPE" name="TYPE"/>
    </xsd:complexType>
    <xsd:complexType name="VEHICLE-CONNECTOR-PIN-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="2" minOccurs="1" name="VEHICLE-CONNECTOR-PIN-
REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="VEHICLE-CONNECTOR-PINS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="VEHICLE-
CONNECTOR-PIN" name="VEHICLE-CONNECTOR-PIN"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="VEHICLE-CONNECTORS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="VEHICLE-
CONNECTOR" name="VEHICLE-CONNECTOR"/>
        </xsd:sequence>
    </xsd:complexType>
```

```
    <xsd:complexType abstract="false" name="VEHICLE-INFORMATION">
        <!--Class: VEHICLE-INFORMATION-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element type="INFO-COMPONENT-REFS" name="INFO-COMPONENT-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="VEHICLE-CONNECTORS" name="VEHICLE-CONNECTORS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="LOGICAL-LINKS" name="LOGICAL-LINKS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="ECU-GROUPS" name="ECU-GROUPS" minOccurs="0"
maxOccurs="1"/>
            <xsd:element type="PHYSICAL-VEHICLE-LINKS" name="PHYSICAL-VEHICLE-
LINKS" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute use="optional" type="xsd:string" name="OID"/>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VEHICLE-INFORMATION-CONNECTOR">
        <!--Class: VEHICLE-INFORMATION-CONNECTOR-->
        <xsd:sequence>
            <xsd:element type="ECU-VARIANT-REFS" name="ECU-VARIANT-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="BASE-VARIANT-REFS" name="BASE-VARIANT-REFS"
minOccurs="0" maxOccurs="1"/>
            <xsd:element type="VEHICLE-INFORMATION-REFS" name="VEHICLE-
INFORMATION-REFS" minOccurs="0" maxOccurs="1"/>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="0" type="DIAG-OBJECT-
CONNECTOR" name="DIAG-OBJECT-CONNECTOR"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="DIAG-OBJECT-
CONNECTOR-REF" type="ODXLINK"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="VEHICLE-INFORMATION-CONNECTORS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="VEHICLE-
INFORMATION-CONNECTOR" name="VEHICLE-INFORMATION-CONNECTOR"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="VEHICLE-INFORMATION-REFS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="VEHICLE-
INFORMATION-REF" type="ODXLINK"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="VEHICLE-INFORMATIONS">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="VEHICLE-
INFORMATION" name="VEHICLE-INFORMATION"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VEHICLE-INFO-SPEC">
        <!--Class: VEHICLE-INFO-SPEC-->
        <xsd:complexContent>
            <xsd:extension base="ODX-CATEGORY">
                <xsd:sequence>
                    <xsd:element type="INFO-COMPONENTS" name="INFO-COMPONENTS"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="VEHICLE-INFORMATIONS" name="VEHICLE-
INFORMATIONS" minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
```

```
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VEHICLE-MODEL">
        <!--Class: VEHICLE-MODEL-->
        <xsd:complexContent>
            <xsd:extension base="INFO-COMPONENT"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VEHICLE-TYPE">
        <!--Class: VEHICLE-TYPE-->
        <xsd:complexContent>
            <xsd:extension base="INFO-COMPONENT"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="VT">
        <!--Class: VT-->
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute use="optional" type="xsd:string" name="TI"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="WRITE-DIAG-COMM-CONNECTOR">
        <!--Class: WRITE-DIAG-COMM-CONNECTOR-->
        <xsd:sequence>
            <xsd:choice maxOccurs="1" minOccurs="1">
                <xsd:element name="WRITE-DIAG-COMM-REF" type="ODXLINK"/>
                <xsd:element name="WRITE-DIAG-COMM-SNREF" type="SNREF"/>
            </xsd:choice>
            <xsd:element type="WRITE-PARAM-VALUES" name="WRITE-PARAM-VALUES"
minOccurs="0" maxOccurs="1"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="WRITE-DATA-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="WRITE-PARAM-VALUE">
        <!--Class: WRITE-PARAM-VALUE-->
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="PHYS-CONSTANT-VALUE"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="IN-PARAM-IF-SNREF"
type="SNREF"/>
        </xsd:sequence>
        <xsd:attribute use="required" type="xsd:string" name="SEMANTIC"/>
    </xsd:complexType>
    <xsd:complexType name="WRITE-PARAM-VALUES">
        <!-- Automatically generated wrapper element type -->
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="1" type="WRITE-PARAM-
VALUE" name="WRITE-PARAM-VALUE"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType abstract="false" name="XDOC">
        <!--Class: XDOC-->
        <xsd:sequence>
            <xsd:group ref="ELEMENT-ID"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="NUMBER"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="STATE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:dateTime"
name="DATE"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="PUBLISHER"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="URL"/>
            <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="POSITION"/>
```

```
        </xsd:sequence>
    </xsd:complexType>
    <xsd:group name="DIAG-COMM-PROXY">
        <xsd:choice>
            <xsd:choice>
                <xsd:element maxOccurs="1" minOccurs="1" type="DIAG-SERVICE"
name="DIAG-SERVICE"/>
                <xsd:element maxOccurs="1" minOccurs="1" type="SINGLE-ECU-JOB"
name="SINGLE-ECU-JOB"/>
            </xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="1" name="DIAG-COMM-REF"
type="ODXLINK"/>
        </xsd:choice>
    </xsd:group>
    <xsd:group name="DIAG-VARIABLE-PROXY">
        <xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="1" name="DIAG-VARIABLE-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="DIAG-VARIABLE"
name="DIAG-VARIABLE"/>
        </xsd:choice>
    </xsd:group>
    <xsd:group name="DTC-PROXY">
        <xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="1" name="DTC-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="DTC" name="DTC"/>
        </xsd:choice>
    </xsd:group>
    <xsd:group name="ELEMENT-ID">
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="SHORT-NAME">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:pattern value="[a-zA-Z0-9_]{1,128}"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element maxOccurs="1" minOccurs="0" name="LONG-NAME">
                <xsd:complexType>
                    <xsd:simpleContent>
                        <xsd:restriction base="TEXT">
                            <xsd:maxLength value="255"/>
                        </xsd:restriction>
                    </xsd:simpleContent>
                </xsd:complexType>
            </xsd:element>
            <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION"
name="DESC"/>
        </xsd:sequence>
    </xsd:group>
    <xsd:group name="ENV-DATA-PROXY">
        <xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="1" name="ENV-DATA-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="ENV-DATA" name="ENV-
DATA"/>
        </xsd:choice>
    </xsd:group>
    <xsd:group name="ODXLINK-TO-TABLE">
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="TABLE-ROW-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:group>
```

```
    <xsd:group name="ODXLINK-TO-TABLE2">
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-ROW-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:group>
    <xsd:group name="ROW-WRAPPER">
        <xsd:choice>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-ROW-REF"
type="ODXLINK"/>
            <xsd:element maxOccurs="1" minOccurs="1" type="TABLE-ROW"
name="TABLE-ROW"/>
        </xsd:choice>
    </xsd:group>
    <xsd:group name="SNREF-TO-TABLE">
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="TABLE-ROW-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:group>
    <xsd:group name="SNREF-TO-TABLE2">
        <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-SNREF"
type="SNREF"/>
            <xsd:element maxOccurs="1" minOccurs="1" name="TABLE-ROW-SNREF"
type="SNREF"/>
        </xsd:sequence>
    </xsd:group>
    <xsd:attributeGroup name="ODXLINK-ATTR">
        <xsd:attribute use="required" type="xsd:string" name="ID-REF"/>
        <xsd:attribute use="optional" name="DOCREF">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:pattern value="[a-zA-Z0-9_]{1,128}"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute use="optional" type="DOCTYPE" name="DOCTYPE"/>
        <xsd:attribute use="optional" type="xsd:string" name="REVISION"/>
    </xsd:attributeGroup>
    <xsd:element type="ODX" name="ODX"/>
</xsd:schema>
```

D.2 XML Schema of sub-structure DESC (odx-xhtml.xsd)

```
Checkerrules
```

D.3 XML Schema of package catalogue (odx-cc.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <xsd:include schemaLocation="odx-xhtml.xsd"/>
       <xsd:complexType name="CATALOG">
              <xsd:sequence>
                     <xsd:element name="SHORT-NAME" type="xsd:string"/>
                     <xsd:element name="COMPANY-DATAS" minOccurs="0">
                            <xsd:complexType>
                                   <xsd:sequence>
                                          <xsd:element name="COMPANY-DATA"
type="COMPANY-DATA" maxOccurs="unbounded"/>
```

```
                                    </xsd:sequence>
                              </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="ADMIN-DATA" type="ADMIN-DATA"
minOccurs="0"/>
                        <xsd:element name="ABLOCKS" minOccurs="0">
                              <xsd:complexType>
                                    <xsd:sequence>
                                          <xsd:element name="ABLOCK" type="ABLOCK"
maxOccurs="unbounded"/>
                                    </xsd:sequence>
                              </xsd:complexType>
                        </xsd:element>
                  </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="ABLOCK">
                  <xsd:sequence>
                        <xsd:group ref="ELEMENT-ID"/>
                        <xsd:element name="CATEGORY" type="xsd:string"
minOccurs="0"/>
                        <xsd:element name="ADMIN-DATA" type="ADMIN-DATA"
minOccurs="0"/>
                        <xsd:element name="FILES" minOccurs="0">
                              <xsd:complexType>
                                    <xsd:sequence>
                                          <xsd:element name="FILE" type="FILE"
maxOccurs="unbounded"/>
                                    </xsd:sequence>
                              </xsd:complexType>
                        </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="UPD" type="UPD-STATUS" default="UNDEFINED"/>
            </xsd:complexType>
            <xsd:complexType name="FILE">
                  <xsd:simpleContent>
                        <xsd:extension base="xsd:string">
                              <xsd:attribute name="CREATION-DATE" type="xsd:date"
use="optional"/>
                              <xsd:attribute name="CREATOR" type="xsd:string"
use="optional"/>
                              <xsd:attribute name="MIME-TYPE" type="xsd:string"
use="optional"/>
                        </xsd:extension>
                  </xsd:simpleContent>
            </xsd:complexType>
            <xsd:simpleType name="UPD-STATUS">
                  <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="NEW"/>
                        <xsd:enumeration value="CHANGED"/>
                        <xsd:enumeration value="UNCHANGED"/>
                        <xsd:enumeration value="UNUSED"/>
                        <xsd:enumeration value="REUSED"/>
                        <xsd:enumeration value="DELETED"/>
                        <xsd:enumeration value="UNDEFINED"/>
                  </xsd:restriction>
            </xsd:simpleType>


            <!-- ++++++++++ Definition of ADMIN-DATA: ++++++++++ -->
            <xsd:complexType name="ADMIN-DATA">
                  <!--Class: ADMIN-DATA-->
                  <!-- The ADMIN-DATA class contains information about administration
of the specific data. -->
                  <xsd:sequence>
                        <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="LANGUAGE"/>
```

```
                    <xsd:element type="COMPANY-DOC-INFOS" name="COMPANY-DOC-
INFOS" minOccurs="0" maxOccurs="1"/>
                    <xsd:element type="DOC-REVISIONS" name="DOC-REVISIONS"
minOccurs="0" maxOccurs="1"/>
             </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="COMPANY-DOC-INFO">
             <!--Class: COMPANY-DOC-INFO-->
             <!-- The COMPANY-DOC-INFO objects contain company-specific
information about the object the ADMIN-DATA belongs to. They are mainly used for
documentation purposes. -->
             <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="COMPANY-DATA-
REF" type="ODXLINK"/>
                    <xsd:element maxOccurs="1" minOccurs="0" name="TEAM-MEMBER-
REF" type="ODXLINK"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="DOC-LABEL"/>
                    <xsd:element type="SDGS" name="SDGS" minOccurs="0"
maxOccurs="1"/>
             </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="COMPANY-DOC-INFOS">
             <!-- Automatically generated wrapper element type -->
             <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="1"
type="COMPANY-DOC-INFO" name="COMPANY-DOC-INFO"/>
             </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="COMPANY-REVISION-INFO">
             <!--Class: COMPANY-REVISION-INFO-->
             <!-- Company-specific revision information can be stored in COMPANY-
REVISION-INFO objects. -->
             <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="COMPANY-DATA-
REF" type="ODXLINK"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="REVISION-LABEL"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="STATE"/>
             </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="COMPANY-REVISION-INFOS">
             <!-- Automatically generated wrapper element type -->
             <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="1"
type="COMPANY-REVISION-INFO" name="COMPANY-REVISION-INFO"/>
             </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="DOC-REVISIONS">
             <!-- Automatically generated wrapper element type -->
             <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="1" type="DOC-
REVISION" name="DOC-REVISION"/>
             </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="DOC-REVISION">
             <!--Class: DOC-REVISION-->
             <!-- Each DOC-REVISION object describes one revision of the object
the ADMIN-DATA belongs to. -->
             <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="0" name="TEAM-MEMBER-
REF" type="ODXLINK"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="REVISION-LABEL"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="STATE"/>
```

```
                        <xsd:element maxOccurs="1" minOccurs="1" type="xsd:dateTime"
name="DATE"/>
                        <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="TOOL"/>
                        <xsd:element type="COMPANY-REVISION-INFOS" name="COMPANY-
REVISION-INFOS" minOccurs="0" maxOccurs="1"/>
                        <xsd:element type="MODIFICATIONS" name="MODIFICATIONS"
minOccurs="0" maxOccurs="1"/>
                  </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="MODIFICATION">
                  <!--Class: MODIFICATION-->
                  <!-- This class represents a modification comment of the editor of a
ODX database. -->
                  <xsd:sequence>
                        <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="CHANGE"/>
                        <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="REASON"/>
                  </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="MODIFICATIONS">
                  <!-- Automatically generated wrapper element type -->
                  <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1"
type="MODIFICATION" name="MODIFICATION"/>
                  </xsd:sequence>
            </xsd:complexType>



            <!-- ++++++++++ Definition of COMPANY-DATA: ++++++++++ -->
            <xsd:complexType name="COMPANY-DATA">
                  <!--Class: COMPANY-DATA-->
                  <!-- This class wraps all data of a company that is involved as a
process partner in the creation or modification of ODX objects. -->
                  <xsd:sequence>
                        <xsd:group ref="ELEMENT-ID"/>
                        <xsd:element type="ROLES" name="ROLES" minOccurs="0"
maxOccurs="1"/>
                        <xsd:element type="TEAM-MEMBERS" name="TEAM-MEMBERS"
minOccurs="0" maxOccurs="1"/>
                        <xsd:element maxOccurs="1" minOccurs="0" type="COMPANY-
SPECIFIC-INFO" name="COMPANY-SPECIFIC-INFO"/>
                  </xsd:sequence>
                  <xsd:attribute use="required" type="xsd:ID" name="ID"/>
                  <xsd:attribute use="optional" type="xsd:string" name="OID"/>
            </xsd:complexType>
            <xsd:complexType name="COMPANY-SPECIFIC-INFO">
                  <!--Class: COMPANY-SPECIFIC-INFO-->
                  <!-- This class represents company spezific information.Information
that is not associated with a single person may be stored also in the COMPANY-
SPECIFIC-INFO object. -->
                  <xsd:sequence>
                        <xsd:element type="RELATED-DOCS" name="RELATED-DOCS"
minOccurs="0" maxOccurs="1"/>
                        <xsd:element type="SDGS" name="SDGS" minOccurs="0"
maxOccurs="1"/>
                  </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="RELATED-DOC">
                  <!--Class: RELATED-DOC-->
                  <!-- RELATED-DOC objects are used for cross referencing additional
documents. -->
                  <xsd:sequence>
                        <xsd:element maxOccurs="1" minOccurs="1" type="XDOC"
name="XDOC"/>
```

```
                    <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION"
name="DESC"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="RELATED-DOCS">
              <!-- Automatically generated wrapper element type -->
              <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="1"
type="RELATED-DOC" name="RELATED-DOC"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="ROLES">
              <!-- Automatically generated wrapper element type -->
              <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="0"
type="xsd:string" name="ROLE"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="TEAM-MEMBERS">
              <!-- Automatically generated wrapper element type -->
              <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="1" type="TEAM-
MEMBER" name="TEAM-MEMBER"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="TEAM-MEMBER">
              <!--Class: TEAM-MEMBER-->
              <!-- This class represents a team member of a company. -->
              <xsd:sequence>
                    <xsd:group ref="ELEMENT-ID"/>
                    <xsd:element type="ROLES" name="ROLES" minOccurs="0"
maxOccurs="1"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="DEPARTMENT"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="ADDRESS"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="ZIP"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="CITY"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="PHONE"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="FAX"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="EMAIL"/>
              </xsd:sequence>
              <xsd:attribute use="required" type="xsd:ID" name="ID"/>
              <xsd:attribute use="optional" type="xsd:string" name="OID"/>
       </xsd:complexType>
       <xsd:complexType name="XDOC">
              <!--Class: XDOC-->
              <!-- XDOC specifies the reference do the additional document. -->
              <xsd:sequence>
                    <xsd:group ref="ELEMENT-ID"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="NUMBER"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="STATE"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:dateTime"
name="DATE"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="PUBLISHER"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="URL"/>
                    <xsd:element maxOccurs="1" minOccurs="0" type="xsd:string"
name="POSITION"/>
```

```
            </xsd:sequence>
      </xsd:complexType>


      <!-- ++++++++++ Definition of SDGS: ++++++++++ -->
      <xsd:complexType name="SD">
            <!--Class: SD-->
            <!-- The SD class contains a special data. -->
            <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                        <xsd:attribute use="optional" type="xsd:string"
name="SI"/>
                        <xsd:attribute use="optional" type="xsd:string"
name="TI"/>
                  </xsd:extension>
            </xsd:simpleContent>
      </xsd:complexType>
      <xsd:complexType name="SDG">
            <!--Class: SDG-->
            <!-- Special data groups (SDGS) are used to store OEM-specific data
not covered by the ODX data model in a structured form. -->
            <xsd:complexContent>
                  <xsd:extension base="SPECIAL-DATA">
                        <xsd:sequence>
                              <xsd:element maxOccurs="1" minOccurs="0"
type="SDG-CAPTION" name="SDG-CAPTION"/>
                              <xsd:choice maxOccurs="unbounded" minOccurs="0">
                                    <xsd:element maxOccurs="1" minOccurs="1"
type="SDG" name="SDG"/>
                                    <xsd:element maxOccurs="1" minOccurs="1"
type="SD" name="SD"/>
                              </xsd:choice>
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
      <xsd:complexType name="SDG-CAPTION">
            <!--Class: SDG-CAPTION-->
            <!-- An SDG contains an optional CAPTION to describe the SDG's
content, and a list of SDGs and SD objects that contain the special data. -->
            <xsd:sequence>
                  <xsd:group ref="ELEMENT-ID"/>
            </xsd:sequence>
            <xsd:attribute use="required" type="xsd:ID" name="ID"/>
            <xsd:attribute use="optional" type="xsd:string" name="OID"/>
      </xsd:complexType>
      <xsd:complexType name="SDGS">
            <!-- Automatically generated wrapper element type -->
            <xsd:sequence>
                  <xsd:element maxOccurs="unbounded" minOccurs="1" type="SDG"
name="SDG"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="SPECIAL-DATA">
            <!--Class: SPECIAL-DATA-->
            <!-- Wrapper class of the SDG and SD. -->
      </xsd:complexType>


      <!-- ++++++++++ Definition of common elements and attributes: ++++++++++ -
->
      <xsd:complexType name="DESCRIPTION">
            <!--Class: DESCRIPTION-->
            <!-- This class gives the possibility to describe the object. -->
            <xsd:sequence>
                  <xsd:element maxOccurs="unbounded" minOccurs="0" type="p"
name="p"/>
```

```
            </xsd:sequence>
            <xsd:attribute use="optional" type="xsd:string" name="TI"/>
      </xsd:complexType>
      <xsd:simpleType name="DOCTYPE">
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="FLASH"/>
                  <xsd:enumeration value="CONTAINER"/>
                  <xsd:enumeration value="LAYER"/>
                  <xsd:enumeration value="MULTIPLE-ECU-JOB-SPEC"/>
                  <xsd:enumeration value="COMPARAM-SPEC"/>
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:group name="ELEMENT-ID">
            <xsd:sequence>
                  <xsd:element maxOccurs="1" minOccurs="1" type="xsd:string"
name="SHORT-NAME"/>
                  <xsd:element maxOccurs="1" minOccurs="0" type="TEXT"
name="LONG-NAME"/>
                  <xsd:element maxOccurs="1" minOccurs="0" type="DESCRIPTION"
name="DESC"/>
            </xsd:sequence>
      </xsd:group>
      <xsd:complexType name="TEXT">
            <!--Class: TEXT-->
            <!-- This class gives the user the possibility to describe a text .
-->
            <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                        <xsd:attribute use="optional" type="xsd:string"
name="TI"/>
                  </xsd:extension>
            </xsd:simpleContent>
      </xsd:complexType>
      <xsd:complexType name="ODXLINK">
            <xsd:attributeGroup ref="ODXLINK-ATTR"/>
      </xsd:complexType>
      <xsd:attributeGroup name="ODXLINK-ATTR">
            <xsd:attribute use="required" type="xsd:string" name="ID-REF"/>
            <xsd:attribute use="optional" type="xsd:string" name="DOCREF"/>
            <xsd:attribute use="optional" type="DOCTYPE" name="DOCTYPE"/>
            <xsd:attribute use="optional" type="xsd:string" name="REVISION"/>
      </xsd:attributeGroup>

      <!-- ++++++++++ Definition of the root element: ++++++++++ -->
      <xsd:element name="CATALOG" type="CATALOG"/>
</xsd:schema>
```

Annex E
(normative)

COMPARAM-SPECs

E.1 KWP2000OnCan

E.1.1 KWP2000_CPS.odx-c

```
<?xml version="1.0" encoding="UTF-8"?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
       <COMPARAM-SPEC ID="ISO_14230_3_on_ISO_14230_2_CPS">
              <SHORT-NAME>KWP2000_CPS</SHORT-NAME>
              <LONG-NAME>KWP2000 COMPARAMs for K-Line</LONG-NAME>
              <PROT-STACKS>
                     <PROT-STACK ID="ISO_14230_3_on_ISO_14230_2_CPPS">
                            <SHORT-NAME>ISO_14230_3_on_ISO_14230_2_CPPS</SHORT-
NAME>
                            <PDU-PROTOCOL-TYPE>ISO_14230_3_on_ISO_14230_2</PDU-
PROTOCOL-TYPE>
                            <COMPARAM-SUBSET-REFS>
                                   <COMPARAM-SUBSET-REF DOCTYPE="COMPARAM-SUBSET"
DOCREF="ISO_14230_3" ID-REF="ISO_14230_3"/>
                                   <COMPARAM-SUBSET-REF DOCTYPE="COMPARAM-SUBSET"
DOCREF="ISO_14230_2" ID-REF="ISO_14230_2"/>
                                   <COMPARAM-SUBSET-REF DOCTYPE="COMPARAM-SUBSET"
DOCREF="ISO_14230_1_UART" ID-REF="ISO_14230_1_UART"/>
                            </COMPARAM-SUBSET-REFS>
                     </PROT-STACK>
              </PROT-STACKS>
       </COMPARAM-SPEC>
</ODX>
```

E.1.2 ISO_14230_3_CPSS.odx

```
<?xml version="1.0" encoding="UTF-8"?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
       <COMPARAM-SUBSET ID="ISO_14230_3" CATEGORY="APP">
              <SHORT-NAME>ISO_14230_3</SHORT-NAME>
              <COMPARAMS>
                     <COMPARAM ID="CP_EnablePerformanceTest" PARAM-CLASS="COM"
CPTYPE="OPTIONAL" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                            <SHORT-NAME>CP_EnablePerformanceTest</SHORT-NAME>
                            <DESC>
                                   <p>This ComParam will place the tester into a
performance measurement mode.  ComParams such as P1Min, P2Min, Br, Cs will be
tested in this mode.</p>
                            </DESC>
                            <PHYSICAL-DEFAULT-VALUE>OFF</PHYSICAL-DEFAULT-VALUE>
                            <DATA-OBJECT-PROP-REF ID-REF="DOP.OnOffSwitch"/>
                     </COMPARAM>
                     <COMPARAM ID="CP_Loopback" PARAM-CLASS="COM"
CPTYPE="OPTIONAL" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                            <SHORT-NAME>CP_Loopback</SHORT-NAME>
                            <DESC>
```

```
                                    <p>Echo Transmitted messages in the receive
queue. Including periodic messages. Loopback messages must only be sent after
successful transmission of a message. Loopback frames are not subject to message
filtering. </p>
                                    <p>0 (OFF) 1 (ON)</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>OFF</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.OnOffSwitch"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_P2Max" PARAM-CLASS="TIMING"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_P2Max</SHORT-NAME>
                              <DESC>
                                    <p>Timeout in receiving an expected frame after
a successful transmit complete. Also used for multiple ECU responses.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>50</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.P2maxCalculation"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_P2Max_Ecu" PARAM-CLASS="TIMING"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-SOFTWARE">
                              <SHORT-NAME>CP_P2Max_Ecu</SHORT-NAME>
                              <DESC>
                                    <p>Performance requirement for the server to
start with the response message after the reception of a request message
(indicated via N_USData.ind).</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>50</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_1"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_P2Min" PARAM-CLASS="TIMING"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_P2Min</SHORT-NAME>
                              <DESC>
                                    <p>This sets the minimum time between tester
request and ECU responses or two ECU responses. After the request, the interface
shall be capable of handling an immediate response (P2_min=0). For subsequent
responses, a byte received after P1_MAX shall be considered as the start of the
subsequent response.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>25</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_P3Max_Ecu" PARAM-CLASS="TIMING"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-SOFTWARE">
                              <SHORT-NAME>CP_P3Max_Ecu</SHORT-NAME>
                              <DESC>
                                    <p>Time between end of ECU responses and start
of new tester request.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>5000</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_1"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_P3Min" PARAM-CLASS="TIMING"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_P3Min</SHORT-NAME>
                              <DESC>
                                    <p>Minimum time between end of ECU responses and
start of new request. The interface will accept all responses up to P3_MIN time.
The interface will allow transmission of a request any time after P3_MIN.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>55</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_RC21CompletionTimeout" PARAM-CLASS="ERRHDL"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_RC21CompletionTimeout</SHORT-NAME>
```

```
                        <DESC>
                              <p>Time period the tester accepts repeated
negative responses with response code 21 and repeats the same request. </p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_1"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_RC21Handling" PARAM-CLASS="ERRHDL"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RC21Handling</SHORT-NAME>
                        <DESC>
                              <p>Repetition mode in case of response code $7F
XX $21.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>Not enabled</PHYSICAL-DEFAULT-
VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.NRCHandling"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_RC21RequestTime" PARAM-CLASS="ERRHDL"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RC21RequestTime</SHORT-NAME>
                        <DESC>
                              <p>Time between negative response with response
code 21 and the retransmission of the same request.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_1"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_RC23CompletionTimeout" PARAM-CLASS="ERRHDL"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RC23CompletionTimeout</SHORT-NAME>
                        <DESC>
                              <p>Time period the tester accepts repeated
negative responses with response code 23 and repeats the same request </p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_1"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_RC23Handling" PARAM-CLASS="ERRHDL"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RC23Handling</SHORT-NAME>
                        <DESC>
                              <p>Repetition mode in case of response code $7F
XX $23.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>Not enabled</PHYSICAL-DEFAULT-
VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.NRCHandling"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_RC23RequestTime" PARAM-CLASS="ERRHDL"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RC23RequestTime</SHORT-NAME>
                        <DESC>
                              <p>Time between negative response with response
code 23 and the retransmission of the same request.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_1"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_RC78CompletionTimeout" PARAM-CLASS="ERRHDL"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RC78CompletionTimeout</SHORT-NAME>
                        <DESC>
                              <p>Time period the tester accepts repeated
negative responses with response code 78 and waits for a positive response
further on.</p>
                        </DESC>
```

```
                        <PHYSICAL-DEFAULT-VALUE>25000</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_1"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_RC78Handling" PARAM-CLASS="ERRHDL"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RC78Handling</SHORT-NAME>
                        <DESC>
                            <p>Handling of $7F XX $78ResponseTimeout and
$78Repetitions.</p>
                            <p>Suported values: 0 (Handled by PDU-API); 1
(Handled by diagnostic application); 2 (N/A); 3 (RC 78 not enabled)</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>Handled by PDU-API</PHYSICAL-
DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.NRC78Handling"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_RepeatReqCounterApp" PARAM-CLASS="ERRHDL"
CPTYPE="OPTIONAL" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RepeatReqCounterApp</SHORT-NAME>
                        <DESC>
                            <p>This parameter contains a counter to enable a
re-transmission of the last request when either a transmit, receive error, or
timeout with no response is detected. This only applies to the application
layer.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.RepeatReqCounter"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_StartMsgIndEnable" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_StartMsgIndEnable</SHORT-NAME>
                        <DESC>
                            <p>Start Message Indication Enable.  Upon
receiving a first frame of a multi-frame message (ISO15765) or upon receiving a
first byte of a UART message and indication will be set in the RX result item.
No data bytes will accompany the result item.</p>
                            <p>0 = Start Message Indication Disabled; 1 =
Start Message  Indication Enabled</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>OFF</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.OnOffSwitch"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_SuspendQueueOnError" PARAM-CLASS="ERRHDL"
CPTYPE="OPTIONAL" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_SuspendQueueOnError</SHORT-NAME>
                        <DESC>
                            <p>This parameter is used as a temporary
parameter for services that require a positive response before any further Com
Primitives can be executed.</p>
                            <p>0 = Do not suspend Queue; 1 = Suspend Queue
on a Timeout Error or on a non-handled 7F error (not an enabled protocol
parameter).</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>OFF</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.OnOffSwitch"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_TesterPresentAddrMode" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_TesterPresentAddrMode</SHORT-NAME>
                        <DESC>
                            <p>Addressing Mode to be used for Tester
Present. Uses the PhysReqxxx or FuncReqxxx ComParams.</p>
                            <p>0 = Use Physical Addressing for the next
message; 1 = Use Functional Addressing for the next message.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>FUNCTIONAL</PHYSICAL-DEFAULT-
VALUE>
```

```
                        <DATA-OBJECT-PROP-REF ID-
REF="DOP.TesterPresentAddrMode"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_TesterPresentHandling" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_TesterPresentHandling</SHORT-NAME>
                        <DESC>
                            <p>Define tester present message generation
settings.</p>
                            <p>0 = Do not generate Tester present messages;
1 = Generate Tester present.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>Generate Tester present
messages</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-
REF="DOP.TesterPresentHandling"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_TesterPresentMessage" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_TesterPresentMessage</SHORT-NAME>
                        <DESC>
                            <p>Define the tester present Message.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>0000000C000000013E</PHYSICAL-
DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-
REF="DOP.TesterPresentMessage"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_TesterPresentReqRsp" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_TesterPresentReqRsp</SHORT-NAME>
                        <DESC>
                            <p>Define settings for present message
responses.</p>
                            <p>0 = No response required upon a Tester
present message; 1 = A response upon a Tester present message is required.  The
response PDU will be discarded by the VCI (D-PDU-API)</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>ON</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.OnOffSwitch"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_TesterPresentSendType" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_TesterPresentSendType</SHORT-NAME>
                        <DESC>
                            <p>Define settings for the type of tester
present transmits.</p>
                            <p>0 = Send on Periodic; 1=Send on Idle.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>Send on Idle</PHYSICAL-DEFAULT-
VALUE>
                        <DATA-OBJECT-PROP-REF ID-
REF="DOP.TesterPresentSendType"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_TesterPresentTime_Ecu" PARAM-CLASS="TIMING"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-SOFTWARE">
                        <SHORT-NAME>CP_TesterPresentTime_Ecu</SHORT-NAME>
                        <DESC>
                            <p>Time for the server to keep a diagnostic
session other than the default session active while not receiving any diagnostic
request message.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>5000</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_3"/>
                    </COMPARAM>
                </COMPARAMS>
                <DATA-OBJECT-PROPS>
```

```
                    <DATA-OBJECT-PROP ID="DOP.OnOffSwitch">
                           <SHORT-NAME>OnOffSwitch</SHORT-NAME>
                           <LONG-NAME>On-Off-Switch</LONG-NAME>
                           <COMPU-METHOD>
                                  <CATEGORY>TEXTTABLE</CATEGORY>
                                  <COMPU-INTERNAL-TO-PHYS>
                                         <COMPU-SCALES>
                                                <COMPU-SCALE>
                                                       <LOWER-LIMIT>0</LOWER-LIMIT>
                                                       <UPPER-LIMIT>0</UPPER-LIMIT>
                                                       <COMPU-CONST>
                                                              <VT>OFF</VT>
                                                       </COMPU-CONST>
                                                </COMPU-SCALE>
                                                <COMPU-SCALE>
                                                       <LOWER-LIMIT>1</LOWER-LIMIT>
                                                       <UPPER-LIMIT>1</UPPER-LIMIT>
                                                       <COMPU-CONST>
                                                              <VT>ON</VT>
                                                       </COMPU-CONST>
                                                </COMPU-SCALE>
                                         </COMPU-SCALES>
                                  </COMPU-INTERNAL-TO-PHYS>
                           </COMPU-METHOD>
                           <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                  <BIT-LENGTH>32</BIT-LENGTH>
                           </DIAG-CODED-TYPE>
                           <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                    </DATA-OBJECT-PROP>
                    <DATA-OBJECT-PROP ID="DOP.NRCHandling">
                           <SHORT-NAME>NRCHandling</SHORT-NAME>
                           <LONG-NAME>Neg. Response Code Handling</LONG-NAME>
                           <DESC>
                                  <p>Repetition mode in case of negative response
code.</p>
                                  <p>Suported values: 0 (Do not repeat); 1 (Repeat
until RC??CompletionTime is over); 2 (Repeat unlimited); 3 (RC Not enabled).</p>
                           </DESC>
                           <COMPU-METHOD>
                                  <CATEGORY>TEXTTABLE</CATEGORY>
                                  <COMPU-INTERNAL-TO-PHYS>
                                         <COMPU-SCALES>
                                                <COMPU-SCALE>
                                                       <LOWER-LIMIT>0</LOWER-LIMIT>
                                                       <UPPER-LIMIT>0</UPPER-LIMIT>
                                                       <COMPU-CONST>
                                                              <VT>Do not repeat</VT>
                                                       </COMPU-CONST>
                                                </COMPU-SCALE>
                                                <COMPU-SCALE>
                                                       <LOWER-LIMIT>1</LOWER-LIMIT>
                                                       <UPPER-LIMIT>1</UPPER-LIMIT>
                                                       <COMPU-CONST>
                                                              <VT>Repeat until
completion time is over</VT>
                                                       </COMPU-CONST>
                                                </COMPU-SCALE>
                                                <COMPU-SCALE>
                                                       <LOWER-LIMIT>2</LOWER-LIMIT>
                                                       <UPPER-LIMIT>2</UPPER-LIMIT>
                                                       <COMPU-CONST>
                                                              <VT>Repeat
unlimited</VT>
                                                       </COMPU-CONST>
                                                </COMPU-SCALE>
                                                <COMPU-SCALE>
```

```
                                                <LOWER-LIMIT>3</LOWER-LIMIT>
                                                <UPPER-LIMIT>3</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>Not enabled</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                    </COMPU-SCALES>
                              </COMPU-INTERNAL-TO-PHYS>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.TesterPresentMessage">
                        <SHORT-NAME>TesterPresentMessage</SHORT-NAME>
                        <LONG-NAME>Tester Present Message</LONG-NAME>
                        <COMPU-METHOD>
                              <CATEGORY>IDENTICAL</CATEGORY>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="MIN-MAX-LENGTH-TYPE" BASE-
DATA-TYPE="A_BYTEFIELD" TERMINATION="END-OF-PDU">
                              <MAX-LENGTH>20</MAX-LENGTH>
                              <MIN-LENGTH>0</MIN-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_BYTEFIELD"/>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.RepeatReqCounter">
                        <SHORT-NAME>RepeatReqCounter</SHORT-NAME>
                        <LONG-NAME>Repeat Request Counter</LONG-NAME>
                        <COMPU-METHOD>
                              <CATEGORY>IDENTICAL</CATEGORY>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
                        <INTERNAL-CONSTR>
                              <LOWER-LIMIT>0</LOWER-LIMIT>
                              <UPPER-LIMIT>255</UPPER-LIMIT>
                        </INTERNAL-CONSTR>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.P2maxCalculation">
                        <SHORT-NAME>P2maxCalculation</SHORT-NAME>
                        <LONG-NAME>P2max calculation with [ms] as units</LONG-
NAME>
                        <DESC>
                              <p>01 to F0: physical = internal * 25; F1 to FE:
(low nibble of hex value) * 256 * 25</p>
                        </DESC>
                        <COMPU-METHOD>
                              <CATEGORY>LINEAR</CATEGORY>
                              <COMPU-INTERNAL-TO-PHYS>
                                    <COMPU-SCALES>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>1</LOWER-LIMIT>
                                                <UPPER-LIMIT>240</UPPER-
LIMIT>
                                                <COMPU-RATIONAL-COEFFS>
                                                      <COMPU-NUMERATOR>
                                                            <V>0</V>
                                                            <V>25</V>
                                                      </COMPU-NUMERATOR>
                                                </COMPU-RATIONAL-COEFFS>
                                          </COMPU-SCALE>
```

```
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>241</LOWER-
LIMIT>
                                                      <UPPER-LIMIT>254</UPPER-
LIMIT>
                                                      <COMPU-RATIONAL-COEFFS>
                                                            <COMPU-NUMERATOR>
                                                                  <V>-1536000</V>
                                                                  <V>64000</V>
                                                            </COMPU-NUMERATOR>
                                                      </COMPU-RATIONAL-COEFFS>
                                                </COMPU-SCALE>
                                          </COMPU-SCALES>
                                    </COMPU-INTERNAL-TO-PHYS>
                              </COMPU-METHOD>
                              <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                    <BIT-LENGTH>32</BIT-LENGTH>
                              </DIAG-CODED-TYPE>
                              <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
                              <INTERNAL-CONSTR>
                                    <LOWER-LIMIT>0</LOWER-LIMIT>
                                    <UPPER-LIMIT>250000</UPPER-LIMIT>
                              </INTERNAL-CONSTR>
                              <UNIT-REF ID-REF="UNIT.MS"/>
                        </DATA-OBJECT-PROP>
                        <DATA-OBJECT-PROP ID="DOP.TesterPresentHandling">
                              <SHORT-NAME>TesterPresentHandling</SHORT-NAME>
                              <LONG-NAME>Tester Present Handling</LONG-NAME>
                              <COMPU-METHOD>
                                    <CATEGORY>TEXTTABLE</CATEGORY>
                                    <COMPU-INTERNAL-TO-PHYS>
                                          <COMPU-SCALES>
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>0</LOWER-LIMIT>
                                                      <UPPER-LIMIT>0</UPPER-LIMIT>
                                                      <COMPU-CONST>
                                                            <VT>Do not generate
Tester present messages</VT>
                                                      </COMPU-CONST>
                                                </COMPU-SCALE>
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>1</LOWER-LIMIT>
                                                      <UPPER-LIMIT>1</UPPER-LIMIT>
                                                      <COMPU-CONST>
                                                            <VT>Generate Tester
present messages</VT>
                                                      </COMPU-CONST>
                                                </COMPU-SCALE>
                                          </COMPU-SCALES>
                                    </COMPU-INTERNAL-TO-PHYS>
                              </COMPU-METHOD>
                              <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                    <BIT-LENGTH>32</BIT-LENGTH>
                              </DIAG-CODED-TYPE>
                              <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                        </DATA-OBJECT-PROP>
                        <DATA-OBJECT-PROP ID="DOP.NRC78Handling">
                              <SHORT-NAME>NRC78Handling</SHORT-NAME>
                              <LONG-NAME>Neg. Response Code 78 Handling</LONG-NAME>
                              <COMPU-METHOD>
                                    <CATEGORY>TEXTTABLE</CATEGORY>
                                    <COMPU-INTERNAL-TO-PHYS>
                                          <COMPU-SCALES>
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>0</LOWER-LIMIT>
```

```
                                                <UPPER-LIMIT>0</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>Handled by PDU-
API</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>1</LOWER-LIMIT>
                                                <UPPER-LIMIT>1</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>Handled by
diagnostic application</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>2</LOWER-LIMIT>
                                                <UPPER-LIMIT>2</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>N/A</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>3</LOWER-LIMIT>
                                                <UPPER-LIMIT>3</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>Not enabled</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                    </COMPU-SCALES>
                              </COMPU-INTERNAL-TO-PHYS>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.TesterPresentAddrMode">
                        <SHORT-NAME>TesterPresentAddrMode</SHORT-NAME>
                        <LONG-NAME>Tester Present Addressing Mode</LONG-NAME>
                        <COMPU-METHOD>
                              <CATEGORY>TEXTTABLE</CATEGORY>
                              <COMPU-INTERNAL-TO-PHYS>
                                    <COMPU-SCALES>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>0</LOWER-LIMIT>
                                                <UPPER-LIMIT>0</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>PHYSICAL</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>1</LOWER-LIMIT>
                                                <UPPER-LIMIT>1</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>FUNCTIONAL</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                    </COMPU-SCALES>
                              </COMPU-INTERNAL-TO-PHYS>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                  </DATA-OBJECT-PROP>
```

```
                    <DATA-OBJECT-PROP ID="DOP.TesterPresentSendType">
                          <SHORT-NAME>TesterPresentSendType</SHORT-NAME>
                          <LONG-NAME>Tester Present Send-Type</LONG-NAME>
                          <COMPU-METHOD>
                                <CATEGORY>TEXTTABLE</CATEGORY>
                                <COMPU-INTERNAL-TO-PHYS>
                                      <COMPU-SCALES>
                                            <COMPU-SCALE>
                                                  <LOWER-LIMIT>0</LOWER-LIMIT>
                                                  <UPPER-LIMIT>0</UPPER-LIMIT>
                                                  <COMPU-CONST>
                                                        <VT>Send on
Periodic</VT>
                                                  </COMPU-CONST>
                                            </COMPU-SCALE>
                                            <COMPU-SCALE>
                                                  <LOWER-LIMIT>1</LOWER-LIMIT>
                                                  <UPPER-LIMIT>1</UPPER-LIMIT>
                                                  <COMPU-CONST>
                                                        <VT>Send on Idle</VT>
                                                  </COMPU-CONST>
                                            </COMPU-SCALE>
                                      </COMPU-SCALES>
                                </COMPU-INTERNAL-TO-PHYS>
                          </COMPU-METHOD>
                          <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                <BIT-LENGTH>32</BIT-LENGTH>
                          </DIAG-CODED-TYPE>
                          <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                    </DATA-OBJECT-PROP>
                    <DATA-OBJECT-PROP ID="DOP.Resolution0_5ms_1">
                          <SHORT-NAME>Resolution0_5ms_1</SHORT-NAME>
                          <LONG-NAME>Resolution 0.5 with [ms] as units and
internal range 0-200000</LONG-NAME>
                          <COMPU-METHOD>
                                <CATEGORY>LINEAR</CATEGORY>
                                <COMPU-INTERNAL-TO-PHYS>
                                      <COMPU-SCALES>
                                            <COMPU-SCALE>
                                                  <LOWER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                  <UPPER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                  <COMPU-RATIONAL-COEFFS>
                                                        <COMPU-NUMERATOR>
                                                              <V>0</V>
                                                              <V>1</V>
                                                        </COMPU-NUMERATOR>
                                                        <COMPU-DENOMINATOR>
                                                              <V>2</V>
                                                        </COMPU-DENOMINATOR>
                                                  </COMPU-RATIONAL-COEFFS>
                                            </COMPU-SCALE>
                                      </COMPU-SCALES>
                                </COMPU-INTERNAL-TO-PHYS>
                          </COMPU-METHOD>
                          <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                <BIT-LENGTH>32</BIT-LENGTH>
                          </DIAG-CODED-TYPE>
                          <PHYSICAL-TYPE BASE-DATA-TYPE="A_FLOAT32"/>
                          <INTERNAL-CONSTR>
                                <LOWER-LIMIT>0</LOWER-LIMIT>
                                <UPPER-LIMIT>200000</UPPER-LIMIT>
                          </INTERNAL-CONSTR>
                          <UNIT-REF ID-REF="UNIT.MS"/>
```

```
                    </DATA-OBJECT-PROP>
                    <DATA-OBJECT-PROP ID="DOP.Resolution0_5ms_2">
                          <SHORT-NAME>Resolution0_5ms_2</SHORT-NAME>
                          <LONG-NAME>Resolution 0.5 with [ms] as units and
internal range 0-500</LONG-NAME>
                          <COMPU-METHOD>
                                <CATEGORY>LINEAR</CATEGORY>
                                <COMPU-INTERNAL-TO-PHYS>
                                      <COMPU-SCALES>
                                            <COMPU-SCALE>
                                                  <LOWER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                  <UPPER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                  <COMPU-RATIONAL-COEFFS>
                                                        <COMPU-NUMERATOR>
                                                              <V>0</V>
                                                              <V>1</V>
                                                        </COMPU-NUMERATOR>
                                                        <COMPU-DENOMINATOR>
                                                              <V>2</V>
                                                        </COMPU-DENOMINATOR>
                                                  </COMPU-RATIONAL-COEFFS>
                                            </COMPU-SCALE>
                                      </COMPU-SCALES>
                                </COMPU-INTERNAL-TO-PHYS>
                          </COMPU-METHOD>
                          <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                <BIT-LENGTH>32</BIT-LENGTH>
                          </DIAG-CODED-TYPE>
                          <PHYSICAL-TYPE BASE-DATA-TYPE="A_FLOAT32"/>
                          <INTERNAL-CONSTR>
                                <LOWER-LIMIT>0</LOWER-LIMIT>
                                <UPPER-LIMIT>500</UPPER-LIMIT>
                          </INTERNAL-CONSTR>
                          <UNIT-REF ID-REF="UNIT.MS"/>
                    </DATA-OBJECT-PROP>
                    <DATA-OBJECT-PROP ID="DOP.Resolution0_5ms_3">
                          <SHORT-NAME>Resolution0_5ms_3</SHORT-NAME>
                          <LONG-NAME>Resolution 0.5 with [ms] as units and
internal range 0-60000</LONG-NAME>
                          <COMPU-METHOD>
                                <CATEGORY>LINEAR</CATEGORY>
                                <COMPU-INTERNAL-TO-PHYS>
                                      <COMPU-SCALES>
                                            <COMPU-SCALE>
                                                  <LOWER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                  <UPPER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                  <COMPU-RATIONAL-COEFFS>
                                                        <COMPU-NUMERATOR>
                                                              <V>0</V>
                                                              <V>1</V>
                                                        </COMPU-NUMERATOR>
                                                        <COMPU-DENOMINATOR>
                                                              <V>2</V>
                                                        </COMPU-DENOMINATOR>
                                                  </COMPU-RATIONAL-COEFFS>
                                            </COMPU-SCALE>
                                      </COMPU-SCALES>
                                </COMPU-INTERNAL-TO-PHYS>
                          </COMPU-METHOD>
                          <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                <BIT-LENGTH>32</BIT-LENGTH>
```

```
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_FLOAT32"/>
                        <INTERNAL-CONSTR>
                              <LOWER-LIMIT>0</LOWER-LIMIT>
                              <UPPER-LIMIT>60000</UPPER-LIMIT>
                        </INTERNAL-CONSTR>
                        <UNIT-REF ID-REF="UNIT.MS"/>
                  </DATA-OBJECT-PROP>
            </DATA-OBJECT-PROPS>
            <UNIT-SPEC>
                  <UNITS>
                        <UNIT ID="UNIT.MS">
                              <SHORT-NAME>MS</SHORT-NAME>
                              <LONG-NAME>milliseconds</LONG-NAME>
                              <DISPLAY-NAME>ms</DISPLAY-NAME>
                              <FACTOR-SI-TO-UNIT>0.001</FACTOR-SI-TO-UNIT>
                              <PHYSICAL-DIMENSION-REF ID-REF="PHYDIM.Time"/>
                        </UNIT>
                  </UNITS>
                  <PHYSICAL-DIMENSIONS>
                        <PHYSICAL-DIMENSION ID="PHYDIM.Time">
                              <SHORT-NAME>Time</SHORT-NAME>
                              <LONG-NAME>Time</LONG-NAME>
                              <TIME-EXP>1</TIME-EXP>
                        </PHYSICAL-DIMENSION>
                  </PHYSICAL-DIMENSIONS>
            </UNIT-SPEC>
      </COMPARAM-SUBSET>
</ODX>
```

E.1.3 ISO_14230_2_CPSS.odx

```
<?xml version="1.0" encoding="UTF-8"?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
      <COMPARAM-SUBSET ID="ISO_14230_2" CATEGORY="TRANS">
            <SHORT-NAME>ISO_14230_2</SHORT-NAME>
            <COMPARAMS>
                  <COMPARAM ID="CP_5BaudAddressFunc" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_5BaudAddressFunc</SHORT-NAME>
                        <DESC>
                              <p>Value of 5Baud Address in case of functional
addressed communication.The correct baud rate address type (functional/physical)
is selected during execution of a Start Communication Com Primitive based on the
setting of  the CP_RequestAddrMode parameter.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>51</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_5BaudAddressPhys" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_5BaudAddressPhys</SHORT-NAME>
                        <DESC>
                              <p>Value of 5Baud Address in case of physical
addressed communication.The correct baud rate address type (functional/physical)
is selected during execution of a Start Communication Com Primitive based on the
setting of  the CP_RequestAddrMode parameter.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>1</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_5BaudMode" PARAM-CLASS="INIT"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_5BaudMode</SHORT-NAME>
                        <DESC>
```

```
                                    <p>Type of 5 Baud initialization.  This
parameter allows either ISO9141 initialization sequence, ISO9141-2/ISO14230
initialization sequence, or hybrid versions which include only one of the extra
bytes defined for ISO9141-2 and ISO14230.</p>
                                    <p>0 (Init as defined in ISO9141-2 and ISO14230-
4); 1 (ISO9141 init followed by interface sending inverted key byte 2, no
inverted address); 2 (ISO9141 init followed by ECU sending inverted address, no
inverted key byte 2); 3 (Init as defined in ISO9141, no inverted key byte 2 nor
inverted address)</p>
                            </DESC>
                            <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-VALUE>
                            <DATA-OBJECT-PROP-REF ID-REF="DOP.5BaudMode"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_EcuRespSourceAddress" PARAM-CLASS="UID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                            <SHORT-NAME>CP_EcuRespSourceAddress</SHORT-NAME>
                            <DESC>
                                    <p>ECU Source Address response of a non-CAN
message.</p>
                            </DESC>
                            <PHYSICAL-DEFAULT-VALUE>16</PHYSICAL-DEFAULT-VALUE>
                            <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_FuncReqFormatPriorityType" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                            <SHORT-NAME>CP_FuncReqFormatPriorityType</SHORT-NAME>
                            <DESC>
                                    <p>First Header Byte of a non-CAN message for a
functional address transmit.</p>
                            </DESC>
                            <PHYSICAL-DEFAULT-VALUE>192</PHYSICAL-DEFAULT-VALUE>
                            <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_FuncRespFormatPriorityType" PARAM-
CLASS="UID" CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                            <SHORT-NAME>CP_FuncRespFormatPriorityType</SHORT-NAME>
                            <DESC>
                                    <p>First Header Byte of a non-CAN message
received from the ECU for functional addressing.</p>
                            </DESC>
                            <PHYSICAL-DEFAULT-VALUE>192</PHYSICAL-DEFAULT-VALUE>
                            <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_FuncReqTargetAddr" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                            <SHORT-NAME>CP_FuncReqTargetAddr</SHORT-NAME>
                            <DESC>
                                    <p>Second Header Byte of a non-CAN message for a
functional address transmit.</p>
                            </DESC>
                            <PHYSICAL-DEFAULT-VALUE>51</PHYSICAL-DEFAULT-VALUE>
                            <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_FuncRespTargetAddr" PARAM-CLASS="UID"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                            <SHORT-NAME>CP_FuncRespTargetAddr</SHORT-NAME>
                            <DESC>
                                    <p>Second Header Byte of a non-CAN message
received from the ECU for functional addressing.  This information is also used
to fill out the functional lookup table for J1850PWM.</p>
                            </DESC>
                            <PHYSICAL-DEFAULT-VALUE>241</PHYSICAL-DEFAULT-VALUE>
                            <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                        </COMPARAM>
                        <COMPARAM ID="CP_HeaderFormatKW" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                            <SHORT-NAME>CP_HeaderFormatKW</SHORT-NAME>
```

```
                        <DESC>
                              <p>Header Byte configuration for K-Line protocol
(Keyword)This setting is used to properly construct the message header bytes to
complete the PDU. This parameter is not used if the protocol parameter RawMode is
set.Header bytes are constructed following the rules of the protocol
specification. This parameter overrides any keybyte values received from the ECU
during initialization, which could be used for automatic header byte
construction.</p>
                              <p>0 (No Header Bytes); 1 (1 Byte Only) (max
size = $3F); 2 (2 Bytes depends on length) 1st byte = size up to $3F, 2nd byte
not used     else         size > $3F 1st byte does not contain size 2nd byte =
size up to $FF; 3 (2 Bytes always)      1st byte never contains size
      information. 2nd byte = size up to $FF; 4 (3 Bytes Only)   1st byte =
Format with size up to    $3F    2nd byte = target address 3rd byte = source
address; 5 (4 Bytes depends on length)  1st byte = size up to $3F  2nd byte =
target address      3rd byte = source address  4th byte not used.else size > $3F.
      1st byte does not contain size.  2nd byte = target address  3rd byte =
source address      4th byte = size up to $FF; 6 (4 Bytes always) 1st byte never
contains size2nd byte = target address  3rd byte = source address  4th byte = size
up to $FF7 (Ford-9141 Header Format)</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>3 Bytes</PHYSICAL-DEFAULT-
VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.HeaderFormatKW"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_InitializationSettings" PARAM-CLASS="INIT"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_InitializationSettings</SHORT-NAME>
                        <DESC>
                              <p>Set Initialization method. </p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>Fast Init sequence</PHYSICAL-
DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-
REF="DOP.InitializationSettings"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_P1Max" PARAM-CLASS="TIMING"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_P1Max</SHORT-NAME>
                        <DESC>
                              <p>Maximum inter-byte time for ECU Responses.
Interface must be capable of handling a P1_MIN time of 0 ms.After the request,
the interface shall be capable of handling an immediate response (P2_MIN=0).  For
subsequent responses, a byte received after P1_MAX shall be considered as the
start of the subsequent response.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>20</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_P1Min" PARAM-CLASS="TIMING"
CPTYPE="OPTIONAL" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_P1Min</SHORT-NAME>
                        <DESC>
                              <p>This sets the minimum inter-byte time for the
ECU responses.  Application shall not get or set this value.  Interface must be
capable of handling P1_MIN=0.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_P4Max" PARAM-CLASS="TIMING"
CPTYPE="OPTIONAL" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_P4Max</SHORT-NAME>
                        <DESC>
                              <p>Maximum inter-byte time for a tester
request.</p>
                        </DESC>
```

```
                        <PHYSICAL-DEFAULT-VALUE>20</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_P4Min" PARAM-CLASS="TIMING"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_P4Min</SHORT-NAME>
                        <DESC>
                            <p>Minimum inter-byte time for tester
transmits.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>5</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_PhysReqFormatPriorityType" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_PhysReqFormatPriorityType</SHORT-NAME>
                        <DESC>
                            <p>First Header Byte of a non-CAN message for
physical address transmit.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>128</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_PhysRespFormatPriorityType" PARAM-
CLASS="UID" CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_PhysRespFormatPriorityType</SHORT-NAME>
                        <DESC>
                            <p>First Header Byte of a non-CAN message
received from the ECU for physical addressing.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>128</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_PhysReqTargetAddr" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_PhysReqTargetAddr</SHORT-NAME>
                        <DESC>
                            <p>Physical Target Addressing Information used
for correct Message Header Construction.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>16</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_RepeatReqCountTrans" PARAM-CLASS="ERRHDL"
CPTYPE="OPTIONAL" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RepeatReqCountTrans</SHORT-NAME>
                        <DESC>
                            <p>This parameter contains a counter to enable a
re-transmission of the last request when either a transmit, a receive error, or
transport layer timeout is detected.  This applies to the transport layer
only.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.RepeatReqCounter"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_RequestAddrMode" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_RequestAddrMode</SHORT-NAME>
                        <DESC>
                            <p>Addressing Mode to be used for the Com
Primitive.</p>
                            <p>1 = Use Physical Addressing for the request;
2 = Use Functional Addressing for the request.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>PHYSICAL</PHYSICAL-DEFAULT-
VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.AddrMode"/>
```

```
                    </COMPARAM>
                    <COMPARAM ID="CP_TesterSourceAddress" PARAM-CLASS="COM"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_TesterSourceAddress</SHORT-NAME>
                              <DESC>
                                    <p>Source address of transmitted message for
non-CAN messages.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>241</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.1ByteUINT"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_TIdle" PARAM-CLASS="INIT" CPTYPE="STANDARD"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_TIdle</SHORT-NAME>
                              <DESC>
                                    <p>Minimum bus idle time before tester starts
the address byte sequence or the fast init sequence.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>300</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_4"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_TInil" PARAM-CLASS="INIT" CPTYPE="STANDARD"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_TInil</SHORT-NAME>
                              <DESC>
                                    <p>Sets the duration for the low pulse in a fast
initialization sequence.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>25</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_TWup" PARAM-CLASS="INIT" CPTYPE="STANDARD"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_TWup</SHORT-NAME>
                              <DESC>
                                    <p>Sets total duration of the wakeup pulse
(TWUP-TINIL)=high pulse before start communication message.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>50</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_W1Max" PARAM-CLASS="INIT" CPTYPE="STANDARD"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_W1Max</SHORT-NAME>
                              <DESC>
                                    <p>Maximum time from the end of address byte to
start of the synchronization pattern from the ECU.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>300</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_5"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_W1Min" PARAM-CLASS="INIT" CPTYPE="OPTIONAL"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_W1Min</SHORT-NAME>
                              <DESC>
                                    <p>Minimum time from the end of address byte to
start of the synchronization pattern from the ECU.</p>
                              </DESC>
                              <PHYSICAL-DEFAULT-VALUE>60</PHYSICAL-DEFAULT-VALUE>
                              <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_W2Max" PARAM-CLASS="INIT" CPTYPE="STANDARD"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                              <SHORT-NAME>CP_W2Max</SHORT-NAME>
                              <DESC>
                                    <p>Maximum time from the end of the
synchronization pattern to the start of key byte 1.</p>
```

```
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>20</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_5"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_W2Min" PARAM-CLASS="INIT" CPTYPE="OPTIONAL"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_W2Min</SHORT-NAME>
                        <DESC>
                              <p>Minimum time from the end of the
synchronization pattern to the start of key byte 1.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>5</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_W3Max" PARAM-CLASS="INIT" CPTYPE="STANDARD"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_W3Max</SHORT-NAME>
                        <DESC>
                              <p>Maximum time between key byte 1 and key byte
2.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>20</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_5"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_W3Min" PARAM-CLASS="INIT" CPTYPE="OPTIONAL"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_W3Min</SHORT-NAME>
                        <DESC>
                              <p>Minimum time between key byte 1 and key byte
2.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>0</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_W4Max" PARAM-CLASS="INIT" CPTYPE="STANDARD"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_W4Max</SHORT-NAME>
                        <DESC>
                              <p>Maximum time between receiving key byte 2
from the vehicle and the inversion being returned by the interface.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>50</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_5"/>
                  </COMPARAM>
                  <COMPARAM ID="CP_W4Min" PARAM-CLASS="INIT" CPTYPE="STANDARD"
DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                        <SHORT-NAME>CP_W4Min</SHORT-NAME>
                        <DESC>
                              <p>Minimum time between receiving key byte 2
from the vehicle and the inversion being returned by the interface.</p>
                        </DESC>
                        <PHYSICAL-DEFAULT-VALUE>25</PHYSICAL-DEFAULT-VALUE>
                        <DATA-OBJECT-PROP-REF ID-REF="DOP.Resolution0_5ms_2"/>
                  </COMPARAM>
            </COMPARAMS>
            <DATA-OBJECT-PROPS>
                  <DATA-OBJECT-PROP ID="DOP.AddrMode">
                        <SHORT-NAME>AddrMode</SHORT-NAME>
                        <LONG-NAME>Addressing Mode</LONG-NAME>
                        <COMPU-METHOD>
                              <CATEGORY>TEXTTABLE</CATEGORY>
                              <COMPU-INTERNAL-TO-PHYS>
                                    <COMPU-SCALES>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>1</LOWER-LIMIT>
                                                <UPPER-LIMIT>1</UPPER-LIMIT>
                                                <COMPU-CONST>
```

```
                                                <VT>PHYSICAL</VT>
                                          </COMPU-CONST>
                                    </COMPU-SCALE>
                                    <COMPU-SCALE>
                                          <LOWER-LIMIT>2</LOWER-LIMIT>
                                          <UPPER-LIMIT>2</UPPER-LIMIT>
                                          <COMPU-CONST>
                                                <VT>FUNCTIONAL</VT>
                                          </COMPU-CONST>
                                    </COMPU-SCALE>
                              </COMPU-SCALES>
                        </COMPU-INTERNAL-TO-PHYS>
                  </COMPU-METHOD>
                  <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                        <BIT-LENGTH>32</BIT-LENGTH>
                  </DIAG-CODED-TYPE>
                  <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
            </DATA-OBJECT-PROP>
            <DATA-OBJECT-PROP ID="DOP.InitializationSettings">
                  <SHORT-NAME>InitializationSettings</SHORT-NAME>
                  <LONG-NAME>Initialization Settings</LONG-NAME>
                  <COMPU-METHOD>
                        <CATEGORY>TEXTTABLE</CATEGORY>
                        <COMPU-INTERNAL-TO-PHYS>
                              <COMPU-SCALES>
                                    <COMPU-SCALE>
                                          <LOWER-LIMIT>1</LOWER-LIMIT>
                                          <UPPER-LIMIT>1</UPPER-LIMIT>
                                          <COMPU-CONST>
                                                <VT>5 Baud Init
sequence</VT>
                                          </COMPU-CONST>
                                    </COMPU-SCALE>
                                    <COMPU-SCALE>
                                          <LOWER-LIMIT>2</LOWER-LIMIT>
                                          <UPPER-LIMIT>2</UPPER-LIMIT>
                                          <COMPU-CONST>
                                                <VT>Fast Init
sequence</VT>
                                          </COMPU-CONST>
                                    </COMPU-SCALE>
                                    <COMPU-SCALE>
                                          <LOWER-LIMIT>3</LOWER-LIMIT>
                                          <UPPER-LIMIT>3</UPPER-LIMIT>
                                          <COMPU-CONST>
                                                <VT>No Init
sequence</VT>
                                          </COMPU-CONST>
                                    </COMPU-SCALE>
                              </COMPU-SCALES>
                        </COMPU-INTERNAL-TO-PHYS>
                  </COMPU-METHOD>
                  <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                        <BIT-LENGTH>32</BIT-LENGTH>
                  </DIAG-CODED-TYPE>
                  <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
            </DATA-OBJECT-PROP>
            <DATA-OBJECT-PROP ID="DOP.1ByteUINT">
                  <SHORT-NAME>1ByteUINT</SHORT-NAME>
                  <LONG-NAME>1ByteUINT</LONG-NAME>
                  <COMPU-METHOD>
                        <CATEGORY>IDENTICAL</CATEGORY>
                  </COMPU-METHOD>
                  <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
```

```
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
                        <INTERNAL-CONSTR>
                              <LOWER-LIMIT>0</LOWER-LIMIT>
                              <UPPER-LIMIT>255</UPPER-LIMIT>
                        </INTERNAL-CONSTR>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.5BaudMode">
                        <SHORT-NAME>5BaudMode</SHORT-NAME>
                        <LONG-NAME>5BaudMode</LONG-NAME>
                        <DESC>
                              <p>0 (Init as defined in ISO9141-2 and ISO14230-
4); 1 (ISO9141 init followed by interface sending inverted key byte 2, no
inverted address); 2 (ISO9141 init followed by ECU sending inverted address, no
inverted key byte 2); 3 (Init as defined in ISO9141, no inverted key byte 2 nor
inverted address)</p>
                        </DESC>
                        <COMPU-METHOD>
                              <CATEGORY>IDENTICAL</CATEGORY>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
                        <INTERNAL-CONSTR>
                              <LOWER-LIMIT>0</LOWER-LIMIT>
                              <UPPER-LIMIT>3</UPPER-LIMIT>
                        </INTERNAL-CONSTR>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.HeaderFormatKW">
                        <SHORT-NAME>HeaderFormatKW</SHORT-NAME>
                        <LONG-NAME>Header Format of ISO14230</LONG-NAME>
                        <COMPU-METHOD>
                              <CATEGORY>TEXTTABLE</CATEGORY>
                              <COMPU-INTERNAL-TO-PHYS>
                                    <COMPU-SCALES>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>0</LOWER-LIMIT>
                                                <UPPER-LIMIT>0</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>No Header
Bytes</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>1</LOWER-LIMIT>
                                                <UPPER-LIMIT>1</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>1 Byte</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>2</LOWER-LIMIT>
                                                <UPPER-LIMIT>2</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>2 Bytes depends on
length</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>3</LOWER-LIMIT>
                                                <UPPER-LIMIT>3</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>2 Bytes
always</VT>
```

```
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>4</LOWER-LIMIT>
                                                <UPPER-LIMIT>4</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>3 Bytes</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>5</LOWER-LIMIT>
                                                <UPPER-LIMIT>5</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>4 Bytes depends on
length</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>6</LOWER-LIMIT>
                                                <UPPER-LIMIT>6</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>4 Bytes
always</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT>7</LOWER-LIMIT>
                                                <UPPER-LIMIT>7</UPPER-LIMIT>
                                                <COMPU-CONST>
                                                      <VT>Ford-9141 Header
Format</VT>
                                                </COMPU-CONST>
                                          </COMPU-SCALE>
                                    </COMPU-SCALES>
                              </COMPU-INTERNAL-TO-PHYS>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.RepeatReqCounter">
                        <SHORT-NAME>RepeatReqCounter</SHORT-NAME>
                        <LONG-NAME>Repeat Request Counter</LONG-NAME>
                        <COMPU-METHOD>
                              <CATEGORY>IDENTICAL</CATEGORY>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
                        <INTERNAL-CONSTR>
                              <LOWER-LIMIT>0</LOWER-LIMIT>
                              <UPPER-LIMIT>255</UPPER-LIMIT>
                        </INTERNAL-CONSTR>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.Resolution0_5ms_2">
                        <SHORT-NAME>Resolution0_5ms_2</SHORT-NAME>
                        <LONG-NAME>Resolution 0.5 with [ms] as units and
internal range 0-500</LONG-NAME>
                        <COMPU-METHOD>
                              <CATEGORY>LINEAR</CATEGORY>
                              <COMPU-INTERNAL-TO-PHYS>
                                    <COMPU-SCALES>
                                          <COMPU-SCALE>
```

```
                                                <LOWER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                <UPPER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                <COMPU-RATIONAL-COEFFS>
                                                      <COMPU-NUMERATOR>
                                                            <V>0</V>
                                                            <V>1</V>
                                                      </COMPU-NUMERATOR>
                                                      <COMPU-DENOMINATOR>
                                                            <V>2</V>
                                                      </COMPU-DENOMINATOR>
                                                </COMPU-RATIONAL-COEFFS>
                                          </COMPU-SCALE>
                                    </COMPU-SCALES>
                              </COMPU-INTERNAL-TO-PHYS>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_FLOAT32"/>
                        <INTERNAL-CONSTR>
                              <LOWER-LIMIT>0</LOWER-LIMIT>
                              <UPPER-LIMIT>500</UPPER-LIMIT>
                        </INTERNAL-CONSTR>
                        <UNIT-REF ID-REF="UNIT.MS"/>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.Resolution0_5ms_4">
                        <SHORT-NAME>Resolution0_5ms_4</SHORT-NAME>
                        <LONG-NAME>Resolution 0.5 with [ms] as units and
internal range 0-20000</LONG-NAME>
                        <COMPU-METHOD>
                              <CATEGORY>LINEAR</CATEGORY>
                              <COMPU-INTERNAL-TO-PHYS>
                                    <COMPU-SCALES>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                <UPPER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                <COMPU-RATIONAL-COEFFS>
                                                      <COMPU-NUMERATOR>
                                                            <V>0</V>
                                                            <V>1</V>
                                                      </COMPU-NUMERATOR>
                                                      <COMPU-DENOMINATOR>
                                                            <V>2</V>
                                                      </COMPU-DENOMINATOR>
                                                </COMPU-RATIONAL-COEFFS>
                                          </COMPU-SCALE>
                                    </COMPU-SCALES>
                              </COMPU-INTERNAL-TO-PHYS>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_FLOAT32"/>
                        <INTERNAL-CONSTR>
                              <LOWER-LIMIT>0</LOWER-LIMIT>
                              <UPPER-LIMIT>20000</UPPER-LIMIT>
                        </INTERNAL-CONSTR>
                        <UNIT-REF ID-REF="UNIT.MS"/>
                  </DATA-OBJECT-PROP>
                  <DATA-OBJECT-PROP ID="DOP.Resolution0_5ms_5">
                        <SHORT-NAME>Resolution0_5ms_5</SHORT-NAME>
```

```
                        <LONG-NAME>Resolution 0.5 with [ms] as units and
internal range 0-2000</LONG-NAME>
                        <COMPU-METHOD>
                              <CATEGORY>LINEAR</CATEGORY>
                              <COMPU-INTERNAL-TO-PHYS>
                                    <COMPU-SCALES>
                                          <COMPU-SCALE>
                                                <LOWER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                <UPPER-LIMIT INTERVAL-
TYPE="INFINITE"/>
                                                <COMPU-RATIONAL-COEFFS>
                                                      <COMPU-NUMERATOR>
                                                            <V>0</V>
                                                            <V>1</V>
                                                      </COMPU-NUMERATOR>
                                                      <COMPU-DENOMINATOR>
                                                            <V>2</V>
                                                      </COMPU-DENOMINATOR>
                                                </COMPU-RATIONAL-COEFFS>
                                          </COMPU-SCALE>
                                    </COMPU-SCALES>
                              </COMPU-INTERNAL-TO-PHYS>
                        </COMPU-METHOD>
                        <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                              <BIT-LENGTH>32</BIT-LENGTH>
                        </DIAG-CODED-TYPE>
                        <PHYSICAL-TYPE BASE-DATA-TYPE="A_FLOAT32"/>
                        <INTERNAL-CONSTR>
                              <LOWER-LIMIT>0</LOWER-LIMIT>
                              <UPPER-LIMIT>2000</UPPER-LIMIT>
                        </INTERNAL-CONSTR>
                        <UNIT-REF ID-REF="UNIT.MS"/>
                  </DATA-OBJECT-PROP>
            </DATA-OBJECT-PROPS>
            <UNIT-SPEC>
                  <UNITS>
                        <UNIT ID="UNIT.MS">
                              <SHORT-NAME>MS</SHORT-NAME>
                              <LONG-NAME>milliseconds</LONG-NAME>
                              <DISPLAY-NAME>ms</DISPLAY-NAME>
                              <FACTOR-SI-TO-UNIT>0.001</FACTOR-SI-TO-UNIT>
                              <PHYSICAL-DIMENSION-REF ID-REF="PHYDIM.Time"/>
                        </UNIT>
                  </UNITS>
                  <PHYSICAL-DIMENSIONS>
                        <PHYSICAL-DIMENSION ID="PHYDIM.Time">
                              <SHORT-NAME>Time</SHORT-NAME>
                              <LONG-NAME>Time</LONG-NAME>
                              <TIME-EXP>1</TIME-EXP>
                        </PHYSICAL-DIMENSION>
                  </PHYSICAL-DIMENSIONS>
            </UNIT-SPEC>
      </COMPARAM-SUBSET>
</ODX>
```

E.1.4 ISO_14230_1_UART_CPSS.odx

```
<?xml version="1.0" encoding="UTF-8"?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
      <COMPARAM-SUBSET ID="ISO_14230_1_UART" CATEGORY="PHYS">
            <SHORT-NAME>ISO_14230_1_UART</SHORT-NAME>
            <COMPARAMS>
```

```
                    <COMPARAM ID="CP_Baudrate" PARAM-CLASS="BUSTYPE"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                           <SHORT-NAME>CP_Baudrate</SHORT-NAME>
                           <DESC>
                                  <p>Represents the desired baud rate.  If the
desired baud rate cannot be achieved within the tolerance of the protocol, the
interface will remain at the previous baud rate.</p>
                           </DESC>
                           <PHYSICAL-DEFAULT-VALUE>10400</PHYSICAL-DEFAULT-VALUE>
                           <DATA-OBJECT-PROP-REF ID-REF="DOP.4ByteIdenticalBaud"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_K_L_LineInit" PARAM-CLASS="BUSTYPE"
CPTYPE="OPTIONAL" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                           <SHORT-NAME>CP_K_L_LineInit</SHORT-NAME>
                           <DESC>
                                  <p>K and L line usage for ISO9141 and ISO14230
initialization address.</p>
                                  <p>0 (Use L-line and K-line for initialization
address)1 (Use K-line only for initialization address).</p>
                           </DESC>
                           <PHYSICAL-DEFAULT-VALUE>Use L-line and K-line for
initialization address</PHYSICAL-DEFAULT-VALUE>
                           <DATA-OBJECT-PROP-REF ID-REF="DOP.KLLineInit"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_K_LinePullup" PARAM-CLASS="BUSTYPE"
CPTYPE="OPTIONAL" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                           <SHORT-NAME>CP_K_LinePullup</SHORT-NAME>
                           <DESC>
                                  <p>Control the K-Line voltage to either 12V or
24V.</p>
                                  <p>0 (No pull-up); 1 (12V);      2 (24V) </p>
                           </DESC>
                           <PHYSICAL-DEFAULT-VALUE>No pull-up</PHYSICAL-DEFAULT-
VALUE>
                           <DATA-OBJECT-PROP-REF ID-REF="DOP.KLinePullup"/>
                    </COMPARAM>
                    <COMPARAM ID="CP_UartConfig" PARAM-CLASS="BUSTYPE"
CPTYPE="STANDARD" DISPLAY-LEVEL="1" CPUSAGE="ECU-COMM">
                           <SHORT-NAME>CP_UartConfig</SHORT-NAME>
                           <DESC>
                                  <p>Configure the parity, data bit size and stop
bits of a Uart protocol.</p>
                           </DESC>
                           <PHYSICAL-DEFAULT-VALUE>8N1</PHYSICAL-DEFAULT-VALUE>
                           <DATA-OBJECT-PROP-REF ID-REF="DOP.UartConfig"/>
                    </COMPARAM>
             </COMPARAMS>
             <DATA-OBJECT-PROPS>
                    <DATA-OBJECT-PROP ID="DOP.4ByteIdenticalBaud">
                           <SHORT-NAME>4ByteIdenticalBaud</SHORT-NAME>
                           <LONG-NAME>identical value with [baud] as units</LONG-
NAME>
                           <COMPU-METHOD>
                                  <CATEGORY>IDENTICAL</CATEGORY>
                           </COMPU-METHOD>
                           <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                  <BIT-LENGTH>32</BIT-LENGTH>
                           </DIAG-CODED-TYPE>
                           <PHYSICAL-TYPE BASE-DATA-TYPE="A_UINT32"/>
                           <UNIT-REF ID-REF="UNIT.Baud"/>
                    </DATA-OBJECT-PROP>
                    <DATA-OBJECT-PROP ID="DOP.KLLineInit">
                           <SHORT-NAME>KLLineInit</SHORT-NAME>
                           <LONG-NAME>K and L Line Initialization</LONG-NAME>
                           <COMPU-METHOD>
                                  <CATEGORY>TEXTTABLE</CATEGORY>
```

```
                                    <COMPU-INTERNAL-TO-PHYS>
                                          <COMPU-SCALES>
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>0</LOWER-LIMIT>
                                                      <UPPER-LIMIT>0</UPPER-LIMIT>
                                                      <COMPU-CONST>
                                                            <VT>Use L-line and K-
line for initialization address</VT>
                                                      </COMPU-CONST>
                                                </COMPU-SCALE>
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>1</LOWER-LIMIT>
                                                      <UPPER-LIMIT>1</UPPER-LIMIT>
                                                      <COMPU-CONST>
                                                            <VT>Use K-line only
for initialization address</VT>
                                                      </COMPU-CONST>
                                                </COMPU-SCALE>
                                          </COMPU-SCALES>
                                    </COMPU-INTERNAL-TO-PHYS>
                              </COMPU-METHOD>
                              <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                    <BIT-LENGTH>32</BIT-LENGTH>
                              </DIAG-CODED-TYPE>
                              <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                        </DATA-OBJECT-PROP>
                        <DATA-OBJECT-PROP ID="DOP.KLinePullup">
                              <SHORT-NAME>KLinePullup</SHORT-NAME>
                              <LONG-NAME>K Line Pullup</LONG-NAME>
                              <COMPU-METHOD>
                                    <CATEGORY>TEXTTABLE</CATEGORY>
                                    <COMPU-INTERNAL-TO-PHYS>
                                          <COMPU-SCALES>
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>0</LOWER-LIMIT>
                                                      <UPPER-LIMIT>0</UPPER-LIMIT>
                                                      <COMPU-CONST>
                                                            <VT>No pull-up</VT>
                                                      </COMPU-CONST>
                                                </COMPU-SCALE>
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>1</LOWER-LIMIT>
                                                      <UPPER-LIMIT>1</UPPER-LIMIT>
                                                      <COMPU-CONST>
                                                            <VT>12V</VT>
                                                      </COMPU-CONST>
                                                </COMPU-SCALE>
                                                <COMPU-SCALE>
                                                      <LOWER-LIMIT>2</LOWER-LIMIT>
                                                      <UPPER-LIMIT>2</UPPER-LIMIT>
                                                      <COMPU-CONST>
                                                            <VT>24V</VT>
                                                      </COMPU-CONST>
                                                </COMPU-SCALE>
                                          </COMPU-SCALES>
                                    </COMPU-INTERNAL-TO-PHYS>
                              </COMPU-METHOD>
                              <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                                    <BIT-LENGTH>32</BIT-LENGTH>
                              </DIAG-CODED-TYPE>
                              <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
                        </DATA-OBJECT-PROP>
                        <DATA-OBJECT-PROP ID="DOP.UartConfig">
                              <SHORT-NAME>UartConfig</SHORT-NAME>
                              <LONG-NAME>Uart Configuration</LONG-NAME>
```

```
<COMPU-METHOD>
      <CATEGORY>TEXTTABLE</CATEGORY>
      <COMPU-INTERNAL-TO-PHYS>
            <COMPU-SCALES>
                  <COMPU-SCALE>
                        <LOWER-LIMIT>0</LOWER-LIMIT>
                        <UPPER-LIMIT>0</UPPER-LIMIT>
                        <COMPU-CONST>
                              <VT>7N1</VT>
                        </COMPU-CONST>
                  </COMPU-SCALE>
                  <COMPU-SCALE>
                        <LOWER-LIMIT>1</LOWER-LIMIT>
                        <UPPER-LIMIT>1</UPPER-LIMIT>
                        <COMPU-CONST>
                              <VT>7O1</VT>
                        </COMPU-CONST>
                  </COMPU-SCALE>
                  <COMPU-SCALE>
                        <LOWER-LIMIT>2</LOWER-LIMIT>
                        <UPPER-LIMIT>2</UPPER-LIMIT>
                        <COMPU-CONST>
                              <VT>7E1</VT>
                        </COMPU-CONST>
                  </COMPU-SCALE>
                  <COMPU-SCALE>
                        <LOWER-LIMIT>3</LOWER-LIMIT>
                        <UPPER-LIMIT>3</UPPER-LIMIT>
                        <COMPU-CONST>
                              <VT>7N2</VT>
                        </COMPU-CONST>
                  </COMPU-SCALE>
                  <COMPU-SCALE>
                        <LOWER-LIMIT>4</LOWER-LIMIT>
                        <UPPER-LIMIT>4</UPPER-LIMIT>
                        <COMPU-CONST>
                              <VT>7O2</VT>
                        </COMPU-CONST>
                  </COMPU-SCALE>
                  <COMPU-SCALE>
                        <LOWER-LIMIT>5</LOWER-LIMIT>
                        <UPPER-LIMIT>5</UPPER-LIMIT>
                        <COMPU-CONST>
                              <VT>7E2</VT>
                        </COMPU-CONST>
                  </COMPU-SCALE>
                  <COMPU-SCALE>
                        <LOWER-LIMIT>6</LOWER-LIMIT>
                        <UPPER-LIMIT>6</UPPER-LIMIT>
                        <COMPU-CONST>
                              <VT>8N1</VT>
                        </COMPU-CONST>
                  </COMPU-SCALE>
                  <COMPU-SCALE>
                        <LOWER-LIMIT>7</LOWER-LIMIT>
                        <UPPER-LIMIT>7</UPPER-LIMIT>
                        <COMPU-CONST>
                              <VT>8O1</VT>
                        </COMPU-CONST>
                  </COMPU-SCALE>
                  <COMPU-SCALE>
                        <LOWER-LIMIT>8</LOWER-LIMIT>
                        <UPPER-LIMIT>8</UPPER-LIMIT>
                        <COMPU-CONST>
                              <VT>8E1</VT>
                        </COMPU-CONST>
                  </COMPU-SCALE>
```

```
                                        <COMPU-SCALE>
                                               <LOWER-LIMIT>9</LOWER-LIMIT>
                                               <UPPER-LIMIT>9</UPPER-LIMIT>
                                               <COMPU-CONST>
                                                      <VT>8N2</VT>
                                               </COMPU-CONST>
                                        </COMPU-SCALE>
                                        <COMPU-SCALE>
                                               <LOWER-LIMIT>10</LOWER-
LIMIT>
                                               <UPPER-LIMIT>10</UPPER-
LIMIT>
                                               <COMPU-CONST>
                                                      <VT>8O2</VT>
                                               </COMPU-CONST>
                                        </COMPU-SCALE>
                                        <COMPU-SCALE>
                                               <LOWER-LIMIT>11</LOWER-
LIMIT>
                                               <UPPER-LIMIT>11</UPPER-
LIMIT>
                                               <COMPU-CONST>
                                                      <VT>8E2</VT>
                                               </COMPU-CONST>
                                        </COMPU-SCALE>
                                        <COMPU-SCALE>
                                               <LOWER-LIMIT>12</LOWER-
LIMIT>
                                               <UPPER-LIMIT>12</UPPER-
LIMIT>
                                               <COMPU-CONST>
                                                      <VT>9N1</VT>
                                               </COMPU-CONST>
                                        </COMPU-SCALE>
                                        <COMPU-SCALE>
                                               <LOWER-LIMIT>13</LOWER-
LIMIT>
                                               <UPPER-LIMIT>13</UPPER-
LIMIT>
                                               <COMPU-CONST>
                                                      <VT>9O1</VT>
                                               </COMPU-CONST>
                                        </COMPU-SCALE>
                                        <COMPU-SCALE>
                                               <LOWER-LIMIT>14</LOWER-
LIMIT>
                                               <UPPER-LIMIT>14</UPPER-
LIMIT>
                                               <COMPU-CONST>
                                                      <VT>9E1</VT>
                                               </COMPU-CONST>
                                        </COMPU-SCALE>
                                        <COMPU-SCALE>
                                               <LOWER-LIMIT>15</LOWER-
LIMIT>
                                               <UPPER-LIMIT>15</UPPER-
LIMIT>
                                               <COMPU-CONST>
                                                      <VT>9N2</VT>
                                               </COMPU-CONST>
                                        </COMPU-SCALE>
                                        <COMPU-SCALE>
                                               <LOWER-LIMIT>16</LOWER-
LIMIT>
                                               <UPPER-LIMIT>16</UPPER-
LIMIT>
                                               <COMPU-CONST>
```

```
                                                        <VT>9O2</VT>
                                                </COMPU-CONST>
                                        </COMPU-SCALE>
                                        <COMPU-SCALE>
                                                <LOWER-LIMIT>17</LOWER-
LIMIT>
                                                <UPPER-LIMIT>17</UPPER-
LIMIT>
                                                <COMPU-CONST>
                                                        <VT>9E2</VT>
                                                </COMPU-CONST>
                                        </COMPU-SCALE>
                                </COMPU-SCALES>
                        </COMPU-INTERNAL-TO-PHYS>
                </COMPU-METHOD>
                <DIAG-CODED-TYPE xsi:type="STANDARD-LENGTH-TYPE" BASE-
DATA-TYPE="A_UINT32">
                        <BIT-LENGTH>32</BIT-LENGTH>
                </DIAG-CODED-TYPE>
                <PHYSICAL-TYPE BASE-DATA-TYPE="A_UNICODE2STRING"/>
        </DATA-OBJECT-PROP>
    </DATA-OBJECT-PROPS>
    <UNIT-SPEC>
        <UNITS>
            <UNIT ID="UNIT.Baud">
                <SHORT-NAME>Baud</SHORT-NAME>
                <LONG-NAME>Baud</LONG-NAME>
                <DISPLAY-NAME>Baudrate</DISPLAY-NAME>
                <FACTOR-SI-TO-UNIT>1</FACTOR-SI-TO-UNIT>
                <PHYSICAL-DIMENSION-REF ID-
REF="PHYDIM.perTime"/>
            </UNIT>
        </UNITS>
        <PHYSICAL-DIMENSIONS>
            <PHYSICAL-DIMENSION ID="PHYDIM.perTime">
                <SHORT-NAME>perTime</SHORT-NAME>
                <LONG-NAME>per Time</LONG-NAME>
                <TIME-EXP>-1</TIME-EXP>
            </PHYSICAL-DIMENSION>
        </PHYSICAL-DIMENSIONS>
    </UNIT-SPEC>
  </COMPARAM-SUBSET>
</ODX>
```

Annex F
(informative)

ECU-MEM Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ODX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="odx.xsd" MODEL-VERSION="2.1.0">
       <FLASH ID="FLASH_ECU_ID101">
              <SHORT-NAME>FLASH_ECU</SHORT-NAME>
              <ECU-MEMS>
                     <ECU-MEM ID="FLASH_ECU_ID102">
                            <SHORT-NAME>ECUMEM_FLASH_ECU</SHORT-NAME>
                            <MEM>
                                   <SESSIONS>
                                          <SESSION ID="FLASH_ECU_ID110">
                                                 <SHORT-
NAME>FLASH_ECU_SW_V01_ON_HW_V01</SHORT-NAME>
                                                 <LONG-NAME>Progamm Code and Data
Segments with software V01 on hardware V01</LONG-NAME>
                                                 <EXPECTED-IDENTS>
                                                        <EXPECTED-IDENT
ID="FLASH_ECU_ID112">
                                                               <SHORT-
NAME>HardwareVersion</SHORT-NAME>
                                                               <IDENT-VALUES>
                                                                      <IDENT-VALUE
TYPE="A_BYTEFIELD">V01</IDENT-VALUE>
                                                               </IDENT-VALUES>
                                                        </EXPECTED-IDENT>
                                                 </EXPECTED-IDENTS>
                                                 <SECURITYS>
                                                        <SECURITY>
                                                               <SECURITY-METHOD
TYPE="A_ASCIISTRING">SECMETH_SIG01</SECURITY-METHOD>
                                                               <FW-SIGNATURE
TYPE="A_BYTEFIELD">0A5F37</FW-SIGNATURE>
                                                        </SECURITY>
                                                 </SECURITYS>
                                                 <DATABLOCK-REFS>
                                                        <DATABLOCK-REF ID-
REF="FLASH_ECU_ID150"/>
                                                        <DATABLOCK-REF ID-
REF="FLASH_ECU_ID151"/>
                                                        <DATABLOCK-REF ID-
REF="FLASH_ECU_ID153"/>
                                                        <DATABLOCK-REF ID-
REF="FLASH_ECU_ID154"/>
                                                 </DATABLOCK-REFS>
                                          </SESSION>
                                          <SESSION ID="FLASH_ECU_ID115">
                                                 <SHORT-NAME>UPDATE_DATA_V02</SHORT-
NAME>
                                                 <LONG-NAME>Update both Data
Segments to V02</LONG-NAME>
                                                 <CHECKSUMS>
                                                        <CHECKSUM
ID="FLASH_ECU_ID117">
                                                               <SHORT-
NAME>CHECKSUM_CPU1_DB02</SHORT-NAME>

       <FILLBYTE>FF</FILLBYTE>
```

```
                                                        <SOURCE-START-
ADDRESS>010000</SOURCE-START-ADDRESS>
                                                        <CHECKSUM-
ALG>CHECKALGO_01</CHECKSUM-ALG>
                                                        <SOURCE-END-
ADDRESS>0137FF</SOURCE-END-ADDRESS>
                                                        <CHECKSUM-RESULT
TYPE="A_BYTEFIELD">55B0</CHECKSUM-RESULT>
                                                  </CHECKSUM>
                                                  <CHECKSUM
ID="FLASH_ECU_ID118">
                                                        <SHORT-
NAME>CHECKSUM_CPU2_DB02</SHORT-NAME>

       <FILLBYTE>00</FILLBYTE>
                                                        <SOURCE-START-
ADDRESS>6000</SOURCE-START-ADDRESS>
                                                        <CHECKSUM-
ALG>CHECKALGO_02</CHECKSUM-ALG>
                                                        <SOURCE-END-
ADDRESS>7FFF</SOURCE-END-ADDRESS>
                                                        <CHECKSUM-RESULT
TYPE="A_BYTEFIELD">030E</CHECKSUM-RESULT>
                                                  </CHECKSUM>
                                            </CHECKSUMS>
                                            <DATABLOCK-REFS>
                                                  <DATABLOCK-REF ID-
REF="FLASH_ECU_ID163"/>
                                                  <DATABLOCK-REF ID-
REF="FLASH_ECU_ID160"/>
                                            </DATABLOCK-REFS>
                                      </SESSION>
                                      <SESSION ID="FLASH_ECU_ID120">
                                            <SHORT-
NAME>UPDATE_CPU1_HW_V02_WITH_SW_V02</SHORT-NAME>
                                            <LONG-NAME>Update CPU1 Code and
Data Segments to V02</LONG-NAME>
                                            <EXPECTED-IDENTS>
                                                  <EXPECTED-IDENT
ID="FLASH_ECU_ID122">
                                                        <SHORT-
NAME>HardwareVersion</SHORT-NAME>
                                                        <IDENT-VALUES>
                                                              <IDENT-VALUE
TYPE="A_BYTEFIELD">V02</IDENT-VALUE>
                                                        </IDENT-VALUES>
                                                  </EXPECTED-IDENT>
                                            </EXPECTED-IDENTS>
                                            <CHECKSUMS>
                                                  <CHECKSUM
ID="FLASH_ECU_ID123">
                                                        <SHORT-
NAME>CHECKSUM_CPU1_DB02</SHORT-NAME>

       <FILLBYTE>FF</FILLBYTE>
                                                        <SOURCE-START-
ADDRESS>010000</SOURCE-START-ADDRESS>
                                                        <CHECKSUM-
ALG>CHECKALGO_01</CHECKSUM-ALG>
                                                        <SOURCE-END-
ADDRESS>013FFF</SOURCE-END-ADDRESS>
                                                        <CHECKSUM-RESULT
TYPE="A_BYTEFIELD">55B0</CHECKSUM-RESULT>
                                                  </CHECKSUM>
                                            </CHECKSUMS>
                                            <DATABLOCK-REFS>
```

```
                                                <DATABLOCK-REF ID-
REF="FLASH_ECU_ID162"/>
                                                <DATABLOCK-REF ID-
REF="FLASH_ECU_ID163"/>
                                          </DATABLOCK-REFS>
                                    </SESSION>
                                    <SESSION ID="FLASH_ECU_ID124">
                                          <SHORT-
NAME>FLASH_ECU_SW_V02_ON_HW_V02</SHORT-NAME>
                                          <LONG-NAME>Progamm Code and Data
Segments with software V02 on hardware V02</LONG-NAME>
                                          <EXPECTED-IDENTS>
                                                <EXPECTED-IDENT
ID="FLASH_ECU_ID125">
                                                      <SHORT-
NAME>HardwareVersion</SHORT-NAME>
                                                      <IDENT-VALUES>
                                                            <IDENT-VALUE
TYPE="A_BYTEFIELD">V02</IDENT-VALUE>
                                                      </IDENT-VALUES>
                                                </EXPECTED-IDENT>
                                          </EXPECTED-IDENTS>
                                          <SECURITYS>
                                                <SECURITY>
                                                      <SECURITY-METHOD
TYPE="A_ASCIISTRING">SECMETH_SIG01</SECURITY-METHOD>
                                                      <FW-SIGNATURE
TYPE="A_BYTEFIELD">0A5F37</FW-SIGNATURE>
                                                </SECURITY>
                                          </SECURITYS>
                                          <DATABLOCK-REFS>
                                                <DATABLOCK-REF ID-
REF="FLASH_ECU_ID156"/>
                                                <DATABLOCK-REF ID-
REF="FLASH_ECU_ID157"/>
                                                <DATABLOCK-REF ID-
REF="FLASH_ECU_ID159"/>
                                                <DATABLOCK-REF ID-
REF="FLASH_ECU_ID160"/>
                                          </DATABLOCK-REFS>
                                    </SESSION>
                              </SESSIONS>
                              <DATABLOCKS>
                                    <DATABLOCK ID="FLASH_ECU_ID150"
TYPE="CODE">
                                          <SHORT-NAME>CODE_CPU1_V01</SHORT-
NAME>
                                          <FLASHDATA-REF ID-
REF="FLASH_ECU_ID170"/>
                                          <AUDIENCE/>
                                    </DATABLOCK>
                                    <DATABLOCK ID="FLASH_ECU_ID151"
TYPE="DATA">
                                          <SHORT-NAME>DATA_CPU1_V01</SHORT-
NAME>
                                          <FLASHDATA-REF ID-
REF="FLASH_ECU_ID171"/>
                                          <SEGMENTS>
                                                <SEGMENT
ID="FLASH_ECU_ID152">
                                                      <SHORT-
NAME>DATASEG_CPU1_V01</SHORT-NAME>
                                                      <SOURCE-START-
ADDRESS>00</SOURCE-START-ADDRESS>
                                                      <COMPRESSED-
SIZE>15045</COMPRESSED-SIZE>
```

```
SIZE>16384</UNCOMPRESSED-SIZE>


OFFSET">

OFFSET>010000</POSITIVE-OFFSET>



TYPE="CODE">

NAME>

REF="FLASH_ECU_ID172"/>


TYPE="DATA">

NAME>

REF="FLASH_ECU_ID173"/>


ID="FLASH_ECU_ID155">

NAME>DATASEG_CPU2_V01</SHORT-NAME>

ADDRESS>6000</SOURCE-START-ADDRESS>

SIZE>7830</COMPRESSED-SIZE>

SIZE>8192</UNCOMPRESSED-SIZE>



TYPE="CODE">

NAME>

REF="FLASH_ECU_ID174"/>


TYPE="DATA">

NAME>

REF="FLASH_ECU_ID175"/>


ID="FLASH_ECU_ID158">

NAME>DATASEG_CPU1_V02</SHORT-NAME>

ADDRESS>016384</SOURCE-START-ADDRESS>

SIZE>15344</COMPRESSED-SIZE>

SIZE>10000</UNCOMPRESSED-SIZE>
                                        <UNCOMPRESSED-
                                    </SEGMENT>
                                </SEGMENTS>
                                <TARGET-ADDR-OFFSET xsi:type="POS-
                                    <POSITIVE-
                                </TARGET-ADDR-OFFSET>
                                <AUDIENCE/>
                            </DATABLOCK>
                            <DATABLOCK ID="FLASH_ECU_ID153"
                                <SHORT-NAME>CODE_CPU2_V01</SHORT-
                                <FLASHDATA-REF ID-
                                <AUDIENCE/>
                            </DATABLOCK>
                            <DATABLOCK ID="FLASH_ECU_ID154"
                                <SHORT-NAME>DATA_CPU2_V01</SHORT-
                                <FLASHDATA-REF ID-
                                <SEGMENTS>
                                    <SEGMENT
                                        <SHORT-
                                        <SOURCE-START-
                                        <COMPRESSED-
                                        <UNCOMPRESSED-
                                    </SEGMENT>
                                </SEGMENTS>
                                <AUDIENCE/>
                            </DATABLOCK>
                            <DATABLOCK ID="FLASH_ECU_ID156"
                                <SHORT-NAME>CODE_CPU1_V02</SHORT-
                                <FLASHDATA-REF ID-
                                <AUDIENCE/>
                            </DATABLOCK>
                            <DATABLOCK ID="FLASH_ECU_ID157"
                                <SHORT-NAME>DATA_CPU1_V02</SHORT-
                                <FLASHDATA-REF ID-
                                <SEGMENTS>
                                    <SEGMENT
                                        <SHORT-
                                        <SOURCE-START-
                                        <COMPRESSED-
                                        <UNCOMPRESSED-
                                    </SEGMENT>
                                </SEGMENTS>
```

```
                                                    <TARGET-ADDR-OFFSET xsi:type="POS-
OFFSET">
                                                          <POSITIVE-
OFFSET>010000</POSITIVE-OFFSET>
                                                    </TARGET-ADDR-OFFSET>
                                                    <AUDIENCE/>
                                              </DATABLOCK>
                                              <DATABLOCK ID="FLASH_ECU_ID159"
TYPE="CODE">
                                                    <SHORT-NAME>CODE_CPU2_V02</SHORT-
NAME>
                                                    <FLASHDATA-REF ID-
REF="FLASH_ECU_ID176"/>
                                                    <AUDIENCE/>
                                              </DATABLOCK>
                                              <DATABLOCK ID="FLASH_ECU_ID160"
TYPE="DATA">
                                                    <SHORT-NAME>DATA_CPU2_V02</SHORT-
NAME>
                                                    <FLASHDATA-REF ID-
REF="FLASH_ECU_ID177"/>
                                                    <SEGMENTS>
                                                          <SEGMENT
ID="FLASH_ECU_ID161">
                                                                <SHORT-
NAME>DATASEG_CPU2_V02</SHORT-NAME>
                                                                <SOURCE-START-
ADDRESS>6000</SOURCE-START-ADDRESS>
                                                                <COMPRESSED-
SIZE>7703</COMPRESSED-SIZE>
                                                                <UNCOMPRESSED-
SIZE>8192</UNCOMPRESSED-SIZE>
                                                          </SEGMENT>
                                                    </SEGMENTS>
                                                    <AUDIENCE/>
                                              </DATABLOCK>
                                              <DATABLOCK ID="FLASH_ECU_ID162"
TYPE="CODE">
                                                    <SHORT-NAME>CODE_CPU1_V02</SHORT-
NAME>
                                                    <FLASHDATA-REF ID-
REF="FLASH_ECU_ID174"/>
                                                    <FILTERS>
                                                          <FILTER xsi:type="ADDRDEF-
FILTER">
                                                                <FILTER-
START>2000</FILTER-START>
                                                                <FILTER-
END>5FFF</FILTER-END>
                                                          </FILTER>
                                                    </FILTERS>
                                                    <SECURITYS>
                                                          <SECURITY>
                                                                <SECURITY-METHOD
TYPE="A_ASCIISTRING">SECMETH_SIG01</SECURITY-METHOD>
                                                                <FW-SIGNATURE
TYPE="A_BYTEFIELD">1F62BA</FW-SIGNATURE>
                                                          </SECURITY>
                                                    </SECURITYS>
                                                    <AUDIENCE/>
                                              </DATABLOCK>
                                              <DATABLOCK ID="FLASH_ECU_ID163"
TYPE="DATA">
                                                    <SHORT-NAME>DATA_CPU1_V02</SHORT-
NAME>
                                                    <FLASHDATA-REF ID-
REF="FLASH_ECU_ID175"/>
```

```
                                                <FILTERS>
                                                      <FILTER xsi:type="SIZEDEF-
FILTER">
                                                            <FILTER-
START>011000</FILTER-START>
                                                            <FILTER-
SIZE>1024</FILTER-SIZE>
                                                      </FILTER>
                                                </FILTERS>
                                                <SEGMENTS>
                                                      <SEGMENT
ID="FLASH_ECU_ID164">
                                                            <SHORT-
NAME>DATASEG_CPU1_V02</SHORT-NAME>
                                                            <SOURCE-START-
ADDRESS>010000</SOURCE-START-ADDRESS>
                                                            <SOURCE-END-
ADDRESS>013FFF</SOURCE-END-ADDRESS>
                                                      </SEGMENT>
                                                </SEGMENTS>
                                                <AUDIENCE/>
                                          </DATABLOCK>
                                    </DATABLOCKS>
                                    <FLASHDATAS>
                                          <FLASHDATA ID="FLASH_ECU_ID170"
xsi:type="INTERN-FLASHDATA">
                                                <SHORT-
NAME>FLASHDATA_CPU1_01</SHORT-NAME>
                                                <DATAFORMAT SELECTION="INTEL-HEX"/>
                                                <DATA>
                                          :020000020000FC

      :100000003821F64FB80000003800BF193800000052

      :100010003821F64FB80000003821F64FB800000034
                                                ...
                                    </DATA>
                                          </FLASHDATA>
                                          <FLASHDATA ID="FLASH_ECU_ID171"
xsi:type="INTERN-FLASHDATA">
                                                <SHORT-
NAME>FLASHDATA_CPU1_02</SHORT-NAME>
                                                <DATAFORMAT SELECTION="BINARY"/>
                                                <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">7F3F</ENCRYPT-COMPRESS-METHOD>
                                                <DATA>
                                          F64FB80000003800BF193800000052
                                          F64FB80000003821F64FB800000034
                                          F64FB80000003880BC81380000004D
                                                ...
                                    </DATA>
                                          </FLASHDATA>
                                          <FLASHDATA ID="FLASH_ECU_ID172"
xsi:type="INTERN-FLASHDATA">
                                                <SHORT-
NAME>FLASHDATA_CPU2_01</SHORT-NAME>
                                                <DATAFORMAT SELECTION="MOTOROLA-
S"/>
                                                <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
                                                <DATA>
                                    FC
                                    F64FB80000003800BF193800000052
                                    F64FB80000003821F64FB800000034
                                                ...
                                    </DATA>
                                          </FLASHDATA>
```

```
                                        <FLASHDATA ID="FLASH_ECU_ID173"
xsi:type="EXTERN-FLASHDATA">
                                              <SHORT-
NAME>FLASHDATA_CPU2_02</SHORT-NAME>
                                              <DATAFORMAT SELECTION="BINARY"/>
                                              <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
                                              <DATAFILE LATEBOUND-
DATAFILE="true">FLASHDATA_03.bin</DATAFILE>
                                        </FLASHDATA>
                                        <FLASHDATA ID="FLASH_ECU_ID174"
xsi:type="INTERN-FLASHDATA">
                                              <SHORT-
NAME>FLASHDATA_CPU1_01</SHORT-NAME>
                                              <DATAFORMAT SELECTION="INTEL-HEX"/>
                                              <DATA>
                                        :020000020000FC

      :100000003821F64FB80000003800BF193800000052

      :100010003821F64FB80000003821F64FB800000034
                                              ...
                                  </DATA>
                                        </FLASHDATA>
                                        <FLASHDATA ID="FLASH_ECU_ID175"
xsi:type="INTERN-FLASHDATA">
                                              <SHORT-
NAME>FLASHDATA_CPU1_02</SHORT-NAME>
                                              <DATAFORMAT SELECTION="BINARY"/>
                                              <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">7F3F</ENCRYPT-COMPRESS-METHOD>
                                              <DATA>
                                        F64FB80000003800BF193800000052
                                        F64FB80000003821F64FB800000034
                                        F64FB80000003880BC81380000004D
                                        ...
                                  </DATA>
                                        </FLASHDATA>
                                        <FLASHDATA ID="FLASH_ECU_ID176"
xsi:type="INTERN-FLASHDATA">
                                              <SHORT-
NAME>FLASHDATA_CPU2_01</SHORT-NAME>
                                              <DATAFORMAT SELECTION="MOTOROLA-
S"/>
                                              <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
                                              <DATA>
                                  FC
                                  F64FB80000003800BF193800000052
                                  F64FB80000003821F64FB800000034
                                              ...
                                  </DATA>
                                        </FLASHDATA>
                                        <FLASHDATA ID="FLASH_ECU_ID177"
xsi:type="EXTERN-FLASHDATA">
                                              <SHORT-
NAME>FLASHDATA_CPU2_02</SHORT-NAME>
                                              <DATAFORMAT SELECTION="BINARY"/>
                                              <ENCRYPT-COMPRESS-METHOD
TYPE="A_BYTEFIELD">4A</ENCRYPT-COMPRESS-METHOD>
                                              <DATAFILE LATEBOUND-
DATAFILE="true">FLASHDATA_07.bin</DATAFILE>
                                        </FLASHDATA>
                                  </FLASHDATAS>
                            </MEM>
                            <PHYS-MEM ID="FLASH_ECU_ID190">
                                  <SHORT-NAME>FLASH_ECU_PHYS_MEM</SHORT-NAME>
```

```
                                    <PHYS-SEGMENTS>
                                          <PHYS-SEGMENT ID="FLASH_ECU_ID191"
xsi:type="ADDRDEF-PHYS-SEGMENT">
                                                <SHORT-NAME>CPU1_SEGMENT_01</SHORT-
NAME>
                                                <FILLBYTE>FF</FILLBYTE>
                                                <BLOCK-SIZE>1024</BLOCK-SIZE>
                                                <START-ADDRESS>00</START-ADDRESS>
                                                <END-ADDRESS>FFFF</END-ADDRESS>
                                          </PHYS-SEGMENT>
                                          <PHYS-SEGMENT ID="FLASH_ECU_ID192"
xsi:type="ADDRDEF-PHYS-SEGMENT">
                                                <SHORT-NAME>CPU1_SEGMENT_02</SHORT-
NAME>
                                                <FILLBYTE>FF</FILLBYTE>
                                                <BLOCK-SIZE>1024</BLOCK-SIZE>
                                                <START-ADDRESS>010000</START-
ADDRESS>
                                                <END-ADDRESS>013FFF</END-ADDRESS>
                                          </PHYS-SEGMENT>
                                          <PHYS-SEGMENT ID="FLASH_ECU_ID193"
xsi:type="SIZEDEF-PHYS-SEGMENT">
                                                <SHORT-NAME>CPU2_SEGMENT_01</SHORT-
NAME>
                                                <FILLBYTE>00</FILLBYTE>
                                                <BLOCK-SIZE>512</BLOCK-SIZE>
                                                <START-ADDRESS>00</START-ADDRESS>
                                                <SIZE>24576</SIZE>
                                          </PHYS-SEGMENT>
                                          <PHYS-SEGMENT ID="FLASH_ECU_ID194"
xsi:type="SIZEDEF-PHYS-SEGMENT">
                                                <SHORT-NAME>CPU2_SEGMENT_02</SHORT-
NAME>
                                                <FILLBYTE>00</FILLBYTE>
                                                <BLOCK-SIZE>512</BLOCK-SIZE>
                                                <START-ADDRESS>6000</START-ADDRESS>
                                                <SIZE>8192</SIZE>
                                          </PHYS-SEGMENT>
                                    </PHYS-SEGMENTS>
                              </PHYS-MEM>
                        </ECU-MEM>
                  </ECU-MEMS>
                  <ECU-MEM-CONNECTORS>
                        <ECU-MEM-CONNECTOR ID="FLASH_ECU_ID230">
                              <SHORT-NAME>FLASH_ECU_Connector</SHORT-NAME>
                              <SESSION-DESCS>
                                    <SESSION-DESC DIRECTION="DOWNLOAD">
                                          <SHORT-
NAME>DOWNLOAD_SW_V01_COMPLETE</SHORT-NAME>
                                          <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                                          <PRIORITY>0</PRIORITY>
                                          <SESSION-SNREF SHORT-
NAME="FLASH_ECU_SW_V01_ON_HW_V01"/>
                                          <DIAG-COMM-SNREF SHORT-
NAME="DIAGCOMM_FLASH_JOB_01"/>
                                          <AUDIENCE/>
                                    </SESSION-DESC>
                                    <SESSION-DESC DIRECTION="DOWNLOAD">
                                          <SHORT-NAME>UPDATE_DATA_V02</SHORT-NAME>
                                          <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                                          <PRIORITY>50</PRIORITY>
                                          <SESSION-SNREF SHORT-
NAME="UPDATE_DATA_V02"/>
                                          <DIAG-COMM-SNREF SHORT-
NAME="DIAGCOMM_FLASH_JOB_02"/>
                                          <AUDIENCE/>
                                    </SESSION-DESC>
```

```
                                    <SESSION-DESC DIRECTION="DOWNLOAD">
                                          <SHORT-
NAME>UPDATE_CPU1_CODE_AND_DATA_V02</SHORT-NAME>
                                          <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                                          <PRIORITY>10</PRIORITY>
                                          <SESSION-SNREF SHORT-
NAME="UPDATE_CPU1_HW_V02_WITH_SW_V02"/>
                                          <DIAG-COMM-SNREF SHORT-
NAME="DIAGCOMM_FLASH_JOB_03"/>
                                          <AUDIENCE/>
                                    </SESSION-DESC>
                                    <SESSION-DESC DIRECTION="DOWNLOAD">
                                          <SHORT-
NAME>DOWNLOAD_SW_V02_COMPLETE</SHORT-NAME>
                                          <PARTNUMBER>FLASH_ECU_123</PARTNUMBER>
                                          <PRIORITY>0</PRIORITY>
                                          <SESSION-SNREF SHORT-
NAME="FLASH_ECU_SW_V02_ON_HW_V02"/>
                                          <DIAG-COMM-SNREF SHORT-
NAME="DIAGCOMM_FLASH_ECU_JOB_01"/>
                                          <AUDIENCE/>
                                    </SESSION-DESC>
                              </SESSION-DESCS>
                              <ECU-MEM-REF ID-REF="FLASH_ECU_ID001"/>
                        </ECU-MEM-CONNECTOR>
                  </ECU-MEM-CONNECTORS>
            </FLASH>
</ODX>
```

Bibliography

[8] SAE J1587:FEB2002 Electronic Data Interchange Between Microcomputer Systems in Heavy-Duty Vehicle Applications

[9] SAE J1708:AUG2004 Serial Data Communications Between Microcomputer Systems in Heavy-Duty Vehicle Applications

[10] SAE J1850:MAY2001 Class B Data Communications Network Interface.

[11] SAE J1939 Recommended Practice for Serial Control and Communications vehicle Network

[12] SAE J2190 Enhanced E/E Diagnostic Test Modes

[13] SAE J2534-1:DEC2004 Recommended Practice for Pass-Thru Vehicle Programming

[14] SAE J2610:APR2002 Serial Data Communication Interface

[15] ASAM MCD 2, Harmonized Data Objects Version 1.0

[16] IEEE 754, Standard for Binary Floating-Point Arithmetic

Figure directory

Tabledirectory

ASAM e.V.
Arnikastraße 2
D-85635 Höhenkirchen
Germany

Tel.: (+49) 08102 / 8953 17
Fax.: (+49) 08102 / 8953 10
E-mail: info@asam.net
Internet: www.asam.net