**prostep ivip / VDA Recommendation**

**Vehicle Electric Container (VEC)**

Version 1.2

Status: RC2





© prostep ivip Association / VDA

Disclaimer

VDA / prostep ivip Recommendations (VDA / PSI Recommendations) are recommendations that are available for general use. Anyone using these recommendations is responsible for ensuring that they are used correctly. This VDA / PSI Recommendation gives due consideration to the prevailing state-of-the-art at the time of publication. Anyone using VDA / PSI Recommendations must assume responsibility for his or her actions and acts at her/his own risk. The prostep ivip Association, the VDA and the parties involved in drawing up the VDA / PSI Recommendation assume no liability whatsoever. We request that anyone encountering an error or the possibility of an incorrect interpretation when using the VDA / PSI Recommendation should contact the prostep ivip Association (psiissues@prostep.com) immediately so that any errors can be rectified.

Copyright

1. All rights to this VDA/PSI Recommendation, in particular the copyright rights of use, and sale such as the right to duplicate, distribute or publish the recommendation, remain exclusively with the prostep ivip Association and the VDA as well as their members.
2. This VDA/PSI Recommendation may be duplicated and distributed unchanged, for instance for use in the context of creating software or services.
3. It is not permitted to change or edit this VDA/PSI Recommendation.
4. A suitable notice indicating the copyright owner and the restrictions on use must always appear.

Contents

[Disclaimer 2](#_Toc33620261)

[Copyright 2](#_Toc33620262)

[Contents 3](#_Toc33620263)

[List of figures 5](#_Toc33620264)

[1 General 9](#_Toc33620265)

[1.1 Preamble 9](#_Toc33620266)

[1.2 Objectives of the recommendation 9](#_Toc33620268)

[1.3 Changes to preceding versions 10](#_Toc33620269)

[1.4 Compatibility to preceding versions 12](#_Toc33620270)

[1.5 Document structure 13](#_Toc33620271)

[1.6 Abbreviations, terms and definitions 13](#_Toc33620272)

[1.7 Reference 13](#_Toc33620273)

[2 Exemplary business use cases 14](#_Toc33620274)

[2.1 UC1: Exchange of components data (part master data) 14](#_Toc33620275)

[2.2 UC2: Exchange of connectivity data 14](#_Toc33620276)

[2.3 UC3: Exchange of geometry data 15](#_Toc33620277)

[2.4 UC4: Exchange of wiring harness data 16](#_Toc33620278)

[2.5 UC5: Exchange of an integrated model of a complete vehicle network 16](#_Toc33620279)

[3 Requirements on the VEC 18](#_Toc33620280)

[3.1 Requirements in the PDM context / addressing PDM data 18](#_Toc33620281)

[3.1.8 Baselines 20](#_Toc33620282)

[3.2 Requirements addressing variance information 20](#_Toc33620283)

[3.3 Requirements out of the physical harness perspective 21](#_Toc33620284)

[3.3.11 Definition and reuse of complex parts 25](#_Toc33620285)

[3.4 Additional requirements out of the component description perspective 25](#_Toc33620286)

[3.5 Additional requirements out of the geometry/topology perspective 27](#_Toc33620287)

[3.6 Additional requirements out of the connectivity perspective 29](#_Toc33620288)

[3.7 Cross-sectional requirements 32](#_Toc33620289)

[3.7.4 Orthogonal grouping concept 32](#_Toc33620290)

[3.7.5 Application Constraints 32](#_Toc33620291)

[3.8 Other / technical requirements 33](#_Toc33620292)

[4 General Guidelines 35](#_Toc33620293)

[4.1 Handling of Identifiers 35](#_Toc33620294)

[4.2 Extension Mechanisms 36](#_Toc33620295)

[4.3 Type Inheritance 36](#_Toc33620296)

[4.4 Default- and Missing-Value Handling 36](#_Toc33620297)

[4.5 Instantiation of Model Structures 37](#_Toc33620298)

[4.6 VEC-Package 38](#_Toc33620299)

[5 VEC Model Description and XML Representation 40](#_Toc33620300)

[5.1 Key Concepts 40](#_Toc33620301)

[5.2 Basic Datatypes 47](#_Toc33620302)

[5.3 PDM Information 54](#_Toc33620303)

[5.4 General Component Data 57](#_Toc33620304)

[5.5 Component Characteristics 66](#_Toc33620305)

[5.6 EE-Components 75](#_Toc33620306)

[5.7 Instances of Components 78](#_Toc33620307)

[5.8 Composite Part Descriptions 87](#_Toc33620308)

[5.9 Topology and Geometry 93](#_Toc33620309)

[5.10 Connectivity 108](#_Toc33620310)

[5.11 External Mapping 116](#_Toc33620311)

[5.12 XML Representation of the Model 118](#_Toc33620312)

[6 Appendix A: Glossary 121](#_Toc33620313)

[7 Appendix B: Data Model Description 122](#_Toc33620314)

[7.1 Module assignment\_groups 122](#_Toc33620315)

[7.2 Module compatibility 125](#_Toc33620316)

[7.3 Module contacting 125](#_Toc33620317)

[7.4 Module core 128](#_Toc33620318)

[7.5 Module coupling 151](#_Toc33620319)

[7.6 Module custom\_properties 155](#_Toc33620320)

[7.7 Module electrical\_parts 158](#_Toc33620321)

[7.8 Module external\_mapping 211](#_Toc33620322)

[7.9 Module geo\_2d 212](#_Toc33620323)

[7.10 Module geo\_3d 218](#_Toc33620324)

[7.11 Module instancing 226](#_Toc33620325)

[7.12 Module instancing\_electrical\_parts 229](#_Toc33620326)

[7.13 Module instancing\_non\_electrical\_parts 246](#_Toc33620327)

[7.14 Module instructions 251](#_Toc33620328)

[7.15 Module local\_geometry 252](#_Toc33620329)

[7.16 Module mapping 254](#_Toc33620330)

[7.17 Module modules 256](#_Toc33620331)

[7.18 Module net 258](#_Toc33620332)

[7.19 Module non\_electrical\_parts 263](#_Toc33620333)

[7.20 Module part\_structure 275](#_Toc33620334)

[7.21 Module part\_usage 276](#_Toc33620335)

[7.22 Module pdm 277](#_Toc33620336)

[7.23 Module physical\_information 286](#_Toc33620337)

[7.24 Module pin\_wire\_mapping 302](#_Toc33620338)

[7.25 Module placeable\_element 303](#_Toc33620339)

[7.26 Module placement 305](#_Toc33620340)

[7.27 Module requirements\_conformance 311](#_Toc33620341)

[7.28 Module routing 312](#_Toc33620342)

[7.29 Module schematic 313](#_Toc33620343)

[7.30 Module signal 321](#_Toc33620344)

[7.31 Module terminal\_pairing 325](#_Toc33620345)

[7.32 Module topology 327](#_Toc33620346)

[7.33 Module usage\_constraint 342](#_Toc33620347)

[7.34 Module usage\_node 344](#_Toc33620348)

[7.35 Module variants 346](#_Toc33620349)

List of figures

[Figure 1: Parts and Documents 40](#_Toc33620350)

[Figure 1: Assignment Group 43](#_Toc33620351)

[Figure 1: Variants 44](#_Toc33620352)

[Figure 1: Variant Structure 45](#_Toc33620353)

[Figure 1: Usage Node 46](#_Toc33620354)

[Figure 1: Usage Constraints 47](#_Toc33620355)

[Figure 1: Localization of Strings 48](#_Toc33620356)

[Figure 1: Numerical Values & Units 49](#_Toc33620357)

[Figure 1: Extensibility with Custom Properties 50](#_Toc33620358)

[Figure 1: Foundation Classes for Physical Properties 51](#_Toc33620359)

[Figure 1: Alias Identifications 52](#_Toc33620360)

[Figure 1: Open and Closed Enumerations 52](#_Toc33620361)

[Figure 1: Open and Closed Pattern Restrictions 53](#_Toc33620362)

[Figure 1: Lifecycle Information 54](#_Toc33620363)

[Figure 1: Baselines 55](#_Toc33620364)

[Figure 1: Item History 56](#_Toc33620365)

[Figure 1: Item Equivalence 57](#_Toc33620366)

[Figure 1: Description of Parts 58](#_Toc33620367)

[Figure 1: General Technical Part 59](#_Toc33620368)

[Figure 1: Supplementary Parts 60](#_Toc33620369)

[Figure 1: Placeable Elements 61](#_Toc33620370)

[Figure 1: Coordinate Systems of Components 62](#_Toc33620371)

[Figure 1: Part Substitutions 64](#_Toc33620372)

[Figure 1: Conformance to Requirements 65](#_Toc33620373)

[Figure 1: Wire 66](#_Toc33620374)

[Figure 1: Terminals 68](#_Toc33620375)

[Figure 1: Wire End Accessory 68](#_Toc33620376)

[Figure 1: Terminal Pairing 69](#_Toc33620377)

[Figure 1: Connector Housings 70](#_Toc33620378)

[Figure 1: Cavity Mapping 71](#_Toc33620379)

[Figure 1: Cavity Seals and Cavity Plugs 72](#_Toc33620380)

[Figure 1: Wire Protections 72](#_Toc33620381)

[Figure 1: Fixings, Cable Ducts, Cable Ties and Similar 73](#_Toc33620382)

[Figure 1: Grommets 74](#_Toc33620383)

[Figure 1: EE-Components 75](#_Toc33620384)

[Figure 1: EE-Component subclasses 76](#_Toc33620385)

[Figure 1: Multi Fuse 76](#_Toc33620386)

[Figure 1: Pinning Information & Pinning Variance 77](#_Toc33620387)

[Figure 1: Instantiation of Components 78](#_Toc33620388)

[Figure 1: Instances of Wires 80](#_Toc33620389)

[Figure 1: Instances of Terminals 81](#_Toc33620390)

[Figure 1: Instances of Connector Housings 81](#_Toc33620391)

[Figure 1: Instances of Cavity Seals and Cavity Plugs 82](#_Toc33620392)

[Figure 1: Instances of EE-Components 82](#_Toc33620393)

[Figure 1: Instances of Wire Protections 83](#_Toc33620394)

[Figure 1: Instances of Fixings, Cable Ducts, Cable Ties and Similar 83](#_Toc33620395)

[Figure 1: Instances of Grommets 84](#_Toc33620396)

[Figure 1: Instances of undefined Components 84](#_Toc33620397)

[Figure 1: Instances of Placeable Components 85](#_Toc33620398)

[Figure 1: Installation Instructions 86](#_Toc33620399)

[Figure 1: Multi-level Part Structure 87](#_Toc33620400)

[Figure 1: Assemblies, Modules and Harness (Configurations) 89](#_Toc33620401)

[Figure 1: Instantiation Approaches 91](#_Toc33620402)

[Figure 1: Harness and Variants 93](#_Toc33620403)

[Figure 1: Topology- and Topology Group Specification 94](#_Toc33620404)

[Figure 1: Topology Zones 95](#_Toc33620405)

[Figure 1: Hierarchical Topologies 96](#_Toc33620406)

[Figure 1: Bending Restrictions 98](#_Toc33620407)

[Figure 1: 2D-Geometry 99](#_Toc33620408)

[Figure 1: 3D-Geometry 101](#_Toc33620409)

[Figure 1: 3D Curves 103](#_Toc33620410)

[Figure 1: Locations 103](#_Toc33620411)

[Figure 1: Placement and Dimensions 104](#_Toc33620412)

[Figure 1: Default Dimensions 106](#_Toc33620413)

[Figure 1: Routing 107](#_Toc33620414)

[Figure 1: Signal Specification 108](#_Toc33620415)

[Figure 1: Net Specification 109](#_Toc33620416)

[Figure 1: Connection Specification 110](#_Toc33620417)

[Figure 1: Wiring Specification 112](#_Toc33620418)

[Figure 1: Contacting Specification 113](#_Toc33620419)

[Figure 1: Coupling Specification 114](#_Toc33620420)

[Figure 1: Wire Grouping Specification 115](#_Toc33620421)

[Figure 1: Pin Wire Mapping 116](#_Toc33620422)

[Figure 1: External Mapping 117](#_Toc33620423)

[Figure 1: VEC-Root 118](#_Toc33620424)

# General

## Preamble

The complexity of today's vehicle electrical systems is constantly growing. A vast variety of options is on the market. Firmly organized and integrated cross-company development processes are essential, combined with powerful, integrated IT infrastructures to support all cross stakeholders.

Against this background, the prostep ivip project group “Vehicle Electrical Systems Workflow Forum” and its predecessors have developed standardised data formats for the uniform description of wiring harnesses and related data. Providing the Harness Description List (KBL, PSI 19/VDA 4964) and supplementing schemas was a leap forward regarding the improvement of car electric development processes and their integration in the development processes for complete vehicles.

However, supporting the whole electric development processes and providing an integrated view on the complete electrical network of a vehicle was not in the scope of the provided specifications. Additional use cases must be addressed. The objective of the prostep ivip project group “Vehicle Electrical Systems Workflow Forum” is to collect these use cases and specify the Vehicle Electric Container (VEC) based on them as the required standardised data format in this context.

The VEC data format specification harmonizes and integrates the already existing solutions with the newly gathered requirements. The VEC data format specification addresses a significantly extended amount of use cases, focussing not only on one single wiring harness but on an electric system as a whole. The VEC data format specification supports a great variety of data exchange use cases all along the electric system development process.

The definition of the VEC was done with a focus on the requirements of the automotive industry. However, it is not restricted to this domain and it is expected that the VEC specification is applicable in aerospace industry and others as well.

## Objectives of the recommendation

This Recommendation contains the specification of the VEC data format with the objective to

* Define a commonly agreed vocabulary and object semantics for the domain of the electrical design process in vehicles
* Facilitate data exchange between development and business partners in the context of physical electric system development and production planning
* Enable tool integration as well as tool-spanning traceability and tool-spanning change management
* Reduce complexity and at the same time increase flexibility by better decoupling tools and data
* Support paperless processes
* Provide a perspective for a solution for requirement in the context of long-term preservation

Concrete use cases are described in chapter 2.

Note: The VEC data format definition is explicitly not intended to be interpreted as a recommendation for the definition of the internal database structure of software tools.

## Changes to preceding versions

Between this Version (1.2) and the direct predecessor (Version 1.1) over 190 individual issues have been addressed. The following section lists the main subjects that have been changed, improved or added. A complete and detailed change history is available in the ECAD Wiki and in the issue tracking system.

Changes that affect the resulting schema in an incompatible way are marked with a “X” in the last column. For more details on compatibility see Chapter 1.4.

|  |  |
| --- | --- |
| Change | Inc. |
| Reorganization of the Model Outline (Chapter 5) |  |
| Added “General Guidelines” for requirements on VEC implementations that are not strictly related to the model structure (Chapter 4). |  |
| Added model documentation to the generated XML Schema files. |  |
| General orthogonal grouping concept to represent functional mappings and requirements (see AssignmentGroup) |  |
| Added concept for the instantiation of topologies. |  |
| Added concept for hierarchical topologies supporting multiple use case (e.g. better traceability between geometry and harness process, splice position optimization, layered segments with a defined inner structure, composite segments, …) |  |
| Added concept for assigning topologies to zones. |  |
| Completely revised the interpretation of Net- & ConnectionSpecification (Architectural Layer & System Schematic) | X |
| Refactoring of the multi-core representation | X |
| Added support for FIT-Rates for components |  |
| Added concept to express conformance with requirements (see RequirementsConformanceSpecification) |  |
| Added concept to define application constraints on instances (e.g. component nodes) (see ApplicationConstraint) |  |
| Added concept for common variant configurations (base inclusion) |  |
| Added concept to define system schematic traceability for directly mated E/E components. |  |
| Added concept to define multicores in their usage (similar to twisted pairs) |  |
| Added concept for traceability between wires and their respective fusing. |  |
| Added concept to define bending restrictions on topologies. |  |
| Added concept to define baselines (well defined sets of ItemVersions) |  |
| Added concept to integrate with the 3D geometries of individual components (e.g. bounding box, |  |
| Added concept for default tolerance definitions |  |
| Added concept for wire addons in connectors. |  |
| Allowed part usage (component instances without part number) in the bill of material. |  |
| Added support for component selection tables. |  |
| Added concepts to support 150% E/E component definitions. |  |
| Added concepts for the description of fuse boxes and other E/E-Components   * internal connectivity * variance of internal connectivity * modularity |  |
| Improved modification tracking / change detection for the digital representation of documents (independent from the approval process in the domain) |  |
| Refactored 3D representation of segments. Dropped current 3D-curve model and replaced it by complete representation of NURBS. | X |
| Added concept for integrated terminals and supplementary components in different contacting situations (e.g. wire fixations) |  |
| Clarification that contact points are free of variance. |  |
| Refactored attributes for compatibility definitions between terminals, plugs, cavities, seals and wires. | X |
| Added concept for flat band wires and flat cores. |  |
| Dropped support for conformance classes. | X |
| Added support for grouping component ports by connector. | X |
| Definition of complex part relations |  |
| Support for complex custom properties und multiple primitive types. |  |
| Added support for hierarchical structures on variant groups and added multiple attributes to the classes in the variant configuration scope. |  |
| Added support for grommets sealed with additional single wire seals. |  |
| Refactored concept for supplementary parts of components in specified locations (e.g. Slots) | X |
| Added support for diodes |  |
| Added support for cable ties |  |
| Added support for multi-fuses |  |

The following list contains all minor changes, that affected schema compatibility.

|  |  |
| --- | --- |
| Refactored and renamed “ContactSystem” to TerminalPairing | X |
| Path mistakenly inherited from ConfigurableElement | X |
| Moved “referenceElement” Association from PartOccurrence to OccurrenceOrUsage | X |
| Refactoring of WireProtectionRole, introduction of TapeRole | X |
| Redefined semantics for ConnectionGroup and NetGroup | X |
| Removed SealingClass and AbrasionResistanceClass (replaced by general concept RobustnessProperties). | X |
| Removed CompatibilityStatement & CompatibilitySpecification | X |
| Refactored modular slot definition (now using indirect references with PartVersion) | X |
| Refactored CopyrightInformation | X |
| Moved attribute TerminalSpecification.angle to WireReception | X |
| Removed Signal from Net-Layer | X |
| Refactoring of SheetOrChapter | X |
| Deprecation of CavityDesign in TerminalReceptionSpecification and CavitySpecification | X |

## Compatibility to preceding versions

Version 1.2 is an extension of version 1.1. Model changes and extensions are guided by the fundamental principle of keeping already implemented concepts downward compatible as far as possible. However, this was not possible in all cases.

Compatibility is defined in the context of this document as the possibility that XML documents created for version 1.1 are still (schema) valid version 1.2 documents. In that sense, incompatible changes will result in schema validation errors if the version 1.1 file uses the affected model elements. Such changes are listed in Chapter 1.3 explicitly.

Additionally, version 1.2 introduces a large amount of open enumerations. As this reduces the degree of freedom in the model it is very likely that version 1.1 VEC files will not validate against the 1.2 strict schema.

Other changes that might be interpreted as incompatible, even without producing schema validation errors, are all improved or clarified documentations, as it might occur that earlier interpretations are now explicitly invalid model interpretations.

All VEC implementations that currently use custom properties for elements that have now (introduced with this version) explicit concepts should be changed accordingly.

## Document structure

Chapter 2 describes some exemplary use cases for the application of the VEC data format.

Chapter 3 breaks down the requirements that the VEC data format specification has to meet.

Chapter 4 explains general concepts and guidelines that apply in a more cross-sectional way and cannot be linked to a specific individual model element.

Chapter 5 explains the meta model of the VEC data format and explains the concrete XML-based syntax of the VEC data format.

Appendix A contains a glossary for the most common abbreviations.

Appendix B contains the detailed meta model specification with a definition of all classes, attributes and relationships in alphabetical arrangement.

## Abbreviations, terms and definitions

See Appendix A for a list of relevant abbreviations, terms and definitions.

## Reference

Further information about this recommendation and related documents and specifications (e.g. the VEC.xsd) are available from

* The VDA and its working party PLM (see http://www.vda.de)
* The prostep ivip Association respectively the project groups VES Workflow Forum and ECAD-implementer forum (see <https://www.prostep.org/en/medialibrary/publications>, <http://ecad-wiki.prostep.org/>)

In addition to that, special reference goes to the recommendation PSI 19 / VDA 4964 Harness Description List (KBL) as previous recommendation.

# Exemplary business use cases

This chapter describes some exemplary business use cases for the application of the VEC data format. The VEC has been designed with these business cases in mind. However, this list shall not be considered as complete or as a restriction of the usage of the VEC.

## UC1: Exchange of components data (part master data)

The VEC data format specification must have a scalable concept for the exchange of part master data. This includes at least the following sub use cases which can appear separately or in any combination:

* UC1.1: Exchange of components data (part master data) with focus on their technical features
  + Example: components data exchange between components supplier and OEM in order to support design engineers to find and choose applicable parts out of a technical perspective
* UC1.2: Exchange of components data (part master data) with focus on meta data
  + Example: components data exchange between components supplier and OEM in order to support design engineers to find and choose applicable parts out of an organisational perspective (e.g. considering approval information, existence of certain usage constraints, …)
* UC1.3: Exchange of components data (part master data) with focus on relational data
  + Example: components data exchange between OEM and a development partner for the definition which cavities, terminals, cavity seals, cavity plugs, and wires are compatible respectively approved
  + Example: Supply tools with data that produce the final wiring harness definition (in combination with a geometry and a connectivity specification as well as inclusive steps like automatic terminal calculation)
* UC1.4: Exchange of components data (part master data) as far as needed for the understanding of a wiring harness description
  + Example: The KBL case

## UC2: Exchange of connectivity data

The VEC data format specification must have a scalable concept for the exchange of (normally configuration based) connectivity data in the context of the car electric system. This includes at least the following sub use cases which can appear separately or in any combination:

* UC2.1: Exchange of architectural data
  + Example: For seamless traceability up to an abstract function level
  + Example: Abstract description of necessary system connections
  + Example: Supply schematics design tools with data in order to facilitate the schematics design process
* UC2.2: Exchange of schematics data
  + Example: Exchange of schematics data between tools for concept tools (tools that allow the evaluation of different electric architectures) and conventional schematics design tools.
  + Example: Supply schematics design tools with data in order to facilitate the wiring design process
  + Example: Supply routing tools with data (in combination with a geometry specification)
* UC2.3 Exchange of wiring data
  + Example: Support data exchange between OEM and a development partner that can be either requested to complete a wiring definition or alternatively provide a final wiring harness definition
  + Example: Supply tools with data that produce the final wiring harness definition (in combination with the necessary components data and a geometry specification as well as inclusive steps like automatic terminal calculation)
  + Example: Supply routing tools with data (in combination with a geometry specification)
  + Example: Supply tools with data that support the tracing of energy paths
  + Example: Supply tools with data that support the service documentation process

## UC3: Exchange of geometry data

The VEC data format specification must have a scalable concept for the exchange of (normally configuration based) geometry data in the context of the car electric system. This includes at least the following sub use cases which can appear separately or in any combination:

* UC3.1: Exchange of topology and relating part placement information
  + Example: Support data exchange between OEM and a development partner who can be either requested to complete a geometry description or alternatively provide a final wiring harness definition
  + Example: Supply tools with data that produce the final wiring harness definition (in combination with the necessary components data and a connectivity specification as well as inclusive steps like automatic terminal calculation)
  + Example: Supply routing tools with data (in combination with a connectivity specification)
  + Example: Supply tools with data that support the service documentation process
  + Example: Supply concept tools with geometry data (tools that allow the evaluation of different electric architectures)
* UC3.2: Exchange of 2D harness drawing data
  + Example: Support layout-based discussions between OEM and a development partner
  + Example: Supply tools with data that have to visualise topology data
* UC3.3: Exchange of 3D wiring harness geometry data
  + Example: Support data exchange between OEM and a development partner respectively harness manufacturer
  + Example: Supply 2D harness drawing tools with data

## UC4: Exchange of wiring harness data

The VEC data format specification must have a scalable concept for the exchange of (normally configuration dependent) wiring harness data. This includes

* The relevant part master data (compare UC1.4)
* The relevant wiring data (compare UC2.3)
* The relevant geometry data (compare UC3.1 - UC3.3)

At least the following sub-use cases have to be supported:

* UC4.1: Exchange of bill of material data (e.g. module-based)
  + Example: Supply tool for cost and/or weight calculation with data
* UC4.2: Exchange of connection list data (e.g. module-based)
  + Example: Supply tools for test application development with data
* UC4.3: Exchange of complete 150% wiring harness data
  + Example: Support data exchange between OEM and harness manufacturer in the context of the processes cost calculation, production planning and change management
  + Example: Supply reporting tools with data
  + Example: Supply EMC-simulation tools with data

## UC5: Exchange of an integrated model of a complete vehicle network

The VEC data format specification must offer a concept for the data exchange of the above-mentioned use cases in an integrated way. This includes traceability links between the different sections of the model.

* UC5.1: Exchange of all wiring harnesses of a vehicle together.
  + Example: Supply customer service diagnosis with relevant information to localize failures.
  + Example: Perform simulations related to the complete vehicle network (e.g. voltage drops)
* UC5.2: Exchange of wiring harness data together with its different geometrical views (e.g. 3D mounting position, form board)
  + Example: Generate 100% variants of the harness geometry to support collision detection in the DMU.
  + Example: Support the inclusion of environment aspects (e.g. heat, crash) in automated validation algorithms.
* UC5.3: Exchange of wiring harness data together with its corresponding system schematic.
  + Example: Supply simulations tools (e.g. EMC) with information about signals and system relevance of concrete wires.
  + Example: Supply engineers and workshop personal with information to allow a better understanding of the role of specific harness elements in the context of the overall system.

# Requirements on the VEC

This chapter summarises the requirements for the VEC data format.

## Requirements in the PDM context / addressing PDM data

### Project reference

Each part / part version description (not part instance description) must be able to express a project reference respectively a special model series reference. The reference is meant to specify the project respectively the model series for which the part has been / is going to be developed.

### Part Usage Constraints

Each part / part version description must be able to optionally express one or more dedicated constraints about the permitted usage of that part. These constraints are meant to be independent of approval information and can address the following aspects

* Vehicle
  + Vehicle project / model series (CarClassificationLevel2)
  + Vehicle derivative (CarClassificationLevel3)
  + Vehicle model (CarClassificationLevel4)
* Technical Equipment (by option code, e.g. country code, for special-purpose vehicles)
* Time constraints
  + Absolute time constraints
  + Constraints addressing the construction phase / model year
  + Constraints addressing the serial number
* Point of use in the context of the electric system

### Approval information (stating a general permission for use)

For each item / item version (item: part or document), the VEC data format specification must have a concept to optionally express the status of approval. The following values are cross-company agreed:

* NotYetApproved
* Approved
* Withdrawn

Beyond that, the VEC data format specification must have a concept to cope with company specific values concerning the approval level. Finally, the VEC data format specification must have a concept to express further details of a certain approval

* A certain identification
* The person (name, company and department) who approved the part
* The date of approval

### Change history

For each item / item version (item: part or document), the VEC data format specification must be able to express the complete change history. This addresses the following aspects

* Creation information
  + The person (name, company and department) who created the item
  + The creation date
* Version / Version history information
  + The version index of each item version
  + The direct predecessor item versions (by item number and item version) regarding the two agreed item version types “Derivation” and “Sequence”
* Change information
  + The one or more change descriptions each specifying one or more of the following aspects
    - The description of an actual change in comparison to the direct predecessor-version(s) in an informal way
    - A certain label (relevant for drawings)
    - The change date
    - The responsible designer
    - The related change order
    - The related change requests
    - The approver of the change

Note: The VEC data format specification must have a concept to express change history information as specified completely in separate which means without any further technical details.

### Conformance with Requirements

The VEC data format specification must have a concept to express conformity (or non-conformity) of parts with certain requirements or specifications. There are various use cases where parts / components are qualified against specific requirements or specifications (e.g. type-examinations, suitability for certain areas of application or manufacturing methods). The result is an explicit statement about the conformity (or non-conformity) of the component.

### External References

For each part the VEC must be able to specify a document reference (by referring document number and version). Normally, such references can be resolved over a period of several years (which is better than a combination of a file path and file name).

In addition, for selective content/elements the VEC must be able to explicitly refer to external files. This can be helpful e.g. for files containing graphically represented installation instructions.

### VEC file-spanning correlations

The VEC data format specification must have a transparent concept that makes it easy for analysing tools to unambiguously calculate correlations between one VEC file (received at a time A) and another VEC file (received at a different time B). Thereby, the following types of correlation are required to be recognisable:

* No correlation
* Extension of information
* Changes

## Baselines

The VEC data format specification must have a concept to group versioned elements (parts & documents) to a valid baseline.

## Requirements addressing variance information

A multi-variant system/product is often described by a set of variant-free (non-variant) building blocks and a set of rules that defines under which conditions these building blocks are contained in a manufactured product. Such a specification is often referred to as a 150%-definition, because it contains more features than any actually manufactured product. The VEC data format shall contain concepts to support this approach in an appropriate way.

To express 150%-definitions it is required to specify variant configuration terms on relevant configurable elements. The variant configuration terms express the conditions for a product configuration, under which the respective element exists. Depending on the process, this configuration mechanism can be attached to different design elements (e.g. connections in a system schematic, topology segments in a harness geometry definition or modules in a customer specific harness (KSK).

Variant configuration terms are based on a set of product properties/features with cross relations (e.g. mutual exclusion, dependency on each other). Together they define the possible variety of the product. The VEC shall not restrict the amount, the naming nor the semantics of these properties/features and the cross relations. Examples for such properties/features are: the product model, the right- or left-hand-driving, the engine fuel type, the body style as well as different equipment features such as seat heating, air conditioning or driver assistance systems.

The VEC data format specification shall support definition of variant and non-variant elements. Non-variant elements are characterized by the fact that their content is either completely or not at all contained in a manufactured product (e.g. a harness module in a customer specific harness, or an atomic component like a connector). Non-variant elements can in turn be part of variant elements. Variant elements are characterized by the fact that their content is dependant on a concrete product configuration and normally only a fraction of the content is contained in a manufactured product (e.g. the description of a customer specific harness).

The following chapters specify some further general requirements concerning the expression of variance information. In addition, the subchapters of the physical harness, component description, topology/geometry, and connectivity perspectives specify

* For which concrete elements the VEC needs to be able to specify variant configuration terms.
* How and for which elements the VEC needs to be able to express non-variant content respectively variant content.

### Variant configuration terms

It is required that the VEC data format specification has a unified concept for the assignment of variant configuration terms. Elements for which the appearance in a configuration is dependent on the same motivation shall be able to refer the same variant configuration term instance.

The VEC data format specification is required to offer two alternative ways for the expression of a variant configuration term which are equivalent out of the standardisation perspective.

* Expression as a character string that meets a standardised syntax that is either directly defined by the VEC data format specification or by related documents.
* Expression as a character string without any limitations by the VEC data format specification.

### Inheritance of variant configuration terms

The VEC data format specification must have a concept to define variant configuration terms that are in common for a set of elements and that are extendable for specific elements. E.g. all elements in a system schematic share a common base variant configuration term. However, some elements can have more restrictive variant configuration terms.

### Vocabulary of variant configuration terms

The VEC data format specification must have a concept that allows the separate / independent definition of the vocabulary (the set of valid literals) that is used for the specification of variant configuration terms. This concept must be free of constraints concerning the number of identifiers as well as the naming of the identifiers. Furthermore, the concept is required to allow the specification of the complete vocabulary or alternatively a selected fraction.

Note: The VEC data format specification is not wanted to standardise the vocabulary itself respectively any identifiers.

The concept for the specification of the vocabulary must allow the definition of identifiers for elementary elements as well as identifiers for grouping elements. Some examples for elementary elements are: “LL” (left-hand drive vehicle), “CAB” (cabriolet) and “ESC” (electronic stability control). Grouping elements must be able to refer various elementary elements. Thereby, the grouping type must be able to be defined. Some examples for an aggregating grouping are: “series” (definition of the properties/features that are included in the standard configuration) and “winter” (a certain equipment package). An example for an exclusive grouping is the vehicle driving type which can be either right-hand driving or left-hand driving.

## Requirements out of the physical harness perspective

Out of the KBL-perspective, the VEC data format specification must enable the description of all data that a manufacturer needs to plan the wiring harness production. This includes the description of single wiring harnesses as well as so called 150%-harness descriptions that may be furthermore structured in several potentially overlapping modules.

### Part descriptions

The VEC data format description must have a concept which allows the specification of the relevant part master data. This includes, but is not restricted to:

* Identification
  + Part number
  + Version identification
  + Company reference
* Reference to relevant documents (e.g. drawings, requirements specifications, …)
* Abbreviation (e.g. a certain string that is generated out of certain attributes)
* Description (e.g. “Connector Housing 5-pin”)
* Material together with a reference to the relevant material reference system (e.g. DIN, VDE, …)
* Colour information together with a reference to the relevant colour reference system
* Mass information (together with the corresponding unit)
* FIT rates
* In case of wires
  + Cross section area
  + Wire type
  + In case of multi core wires an identification for each of the single cores
  + Wire groupings (together with the relevant twisting specification)
* In case of connector housings
  + Coding
  + Structure
    - Slots together with an identification for each slot
    - Cavities together with an identification for each cavity
  + Modular connectors
  + Required addons for wires
* In case of terminals
  + Terminal type
  + Nominal size

### E/E component interfaces

The VEC format specification must have concept to describe the interfaces of E/E components with information that is required within the scope. This includes, but is not restricted to:

* Mechanical / physical interface
  + Connector geometry
  + Pin properties (e.g. plating material)
* Electrical Interface
  + Voltage level (variant dependant)
  + Current profiles (variant dependant)
  + Internal connectivity (variant dependant, but not software controlled)
* Pluggability
  + with the wiring harness
  + between E/E components (e.g. fuse box & fuse)

### Assembly / harness module / harness configuration set parts list

The VEC format specification must have a uniform concept for the unambiguous specification of parts lists (addressing all associated subparts like connector housings, terminals, wires optionally together with the length information for each core, wire protections, fixings, grommets, and their quantity). This is equally relevant for simple assemblies as well as harness modules and complete harness configuration sets. The parts lists have to be version precise.

### Wire list and connection list

For each wire instance (KBL::Wire\_occurrence, KBL::Core\_occurrence), the VEC data format specification must allow the definition of

* The relevant part master data both for the whole wire and for each core
* The wire instance number
* The contact points the cores interconnect (regarding the contacted pluggable terminal, splice terminal or ring terminal, optionally in combination with a contacted cavity)
* The length information for each core together with a length type (e.g. the DMU-length)
* The connections (KBL::connection) the cores realise
* A signal name for each core respectively connection

### Cavity equipment

The VEC concept for 150% wiring harness descriptions must ensure that for concrete configurations the cavity equipment for each cavity of the included connectors can be unambiguously calculated. This addresses the following information

* What cavities are equipped with which terminal(s) and optionally together with which cavity seals. Which wires / cores are connected?
* What cavities are non-plugged or plugged and, in this case, plugged with which plugs?

### Replacements

The VEC data format specification must have a concept which allows the definition of configuration dependant cavity plug replacements. The cavity plug replacement is e.g. needed in the case that a certain configuration results in a certain combination of modules where at least one module defines an equipment for a cavity that belongs to a plugged connector which is part of another module.

### Topology and Routing

The VEC data format specification must have a concept that allows the definition of the harness topology. Furthermore, for each connection (KBL::connection) or wire instance (KBL::Wire\_occurrence, KBL::Core\_occurrence) it must be possible to define a routing which means an obligatory path through the topology. Comparably, it must be possible to specify the placement of all other part instances (fixings, grommets, …) as a defined location on the topology. The VEC must support components that are attached to multiple locations in the topology (e.g. cable ducts, grommets, connectors).

The routing definition concept must offer the possibility to specify the segments that are mandatory even in the case of a calculation of a new routing. In addition to that, the routing concept must offer the possibility to mark certain routing definitions as special. This is intended to offer for instance an optional marking mechanism to enable the distinguishing of man-made routing definitions and calculated ones.

The VEC must provide all information necessary for routing algorithm to create an automatic routing to and from connectors with multiple bundle position points (e.g. HV-connectors). This includes the information which cavity is reachable from which bundle position point.

### Wire and core lengths

The VEC data format specification must have a concept which allows for each wire and core instance the definition of dedicated lengths information each of it in combination with the relevant length type (e.g. the calculated DMU length, a measured value, …)

### Modules and Harness configuration sets

The VEC data format specification must have a concept that allows the definition of modules (KBL::Module) and complete harness configuration sets (KBL::Harness\_configuration) on the basis of a common 150% wiring harness description. Thereby, the definition of harness configuration sets must be alternatively able to base on the aggregation of elementary part instances (like connector housings) or modules or a combination of both.

Modules and Harness configuration sets have parts character which means they have a part number, version index and optionally all kinds of PDM information like any other part.

### Extra requirements addressing variance information

The VEC data format specification must have a concept that enables the assignment of every occurrence, every module (instance) and every harness configuration set (instance) in the context of a 150% wiring harness description with a certain variant configuration term.

Furthermore, the VEC data format specification must have a concept to describe mutually exclusive modules as part of a module family (KBL – module family concept).

Finally, the VEC data format specification must have a concept to describe certain occurrences as mandatory completion parts in dependency of the existence of one or more related modules (KBL - module list concept).

## Definition and reuse of complex parts

The VEC data format specification must have a concept to define complex parts and reuse them in different positions. Wiring harnesses do not exclusively consist of simple components, but also of parts that are assembled themselves. This can range from ready-made cables (e.g. antenna or USB cables) up to complete (sub) harnesses (e.g. automatically manufactured modules or door harnesses used in the left and right door in the same manufactured way).

The reuse concept shall support at least the following aspects:

* Renaming in the context of usage (e.g. connector names, wire numbers)
* Mapping into the topology / geometry of the usage (e.g. right door / left door)
* Correct electrological mapping in the context of the usage (e.g. realized schematic connections & signals are usage dependant).
* Redefinition of properties in the context of the usage (e.g. some ready-made cables with connectors on only one side might have a fixed length and are cut to the correct length in the concrete usage).
* Definition of the installation in the concrete usage (e.g. positioning of connectors, routing of cables)

## Additional requirements out of the component description perspective

### Usage Nodes

The VEC must provide a concept to support usage nodes. Usage nodes are a representation of an abstract position in a vehicle, independent from the concrete perspective (ELOG, GEO, …). Usage nodes are used to provide a consistent naming over different electrical systems and different development branches.

### Supporting the selection of terminals and related parts

In many companies, the selection process of concrete terminals, cavity seals and cavity plugs is done automatically on the basis of an abstract contacting specification and a components library. Against this background, the VEC data format specification must enable the description of all data those component libraries typically contain. At least, the following information must be able to be described:

* Existing compatibility between cavity and terminal
* Existing compatibility between cavity and cavity plug
* Existing compatibility between cavity and wire (type)
* Existing compatibility between terminal and wire (regarding core and insulation)
* Classification of a terminal as multi crimp able
* Material information of terminals (basic material, plating material of terminal reception and wire reception)
* Compatibility of a cavity seal in dependency of cavity, terminal and wire
* The suited contacting method (crimp, weld, …)

Note: The VEC data format specification must enable compatibility relationships to be explicitly defined (alternatively by instances-based or grouping-type-based compatibility statements) as well as implicitly by type specific (cavity, terminal, …) attributes and its values.

### Design relevant attributes

In many companies, the part descriptions within component libraries contain some technical detail information that are relevant for a designer when he has to search and chose a dedicated part with dedicated technical properties. Against this background, the VEC data format specification must enable the description of the most common technical properties that are typically needed for this purpose.

Below, a few examples of such properties are listed. The list contains only properties in addition to the chapters 3.1, 3.3.1 and 3.4.1.

* All parts concerning
  + The robustness against water, oil, petrol, …
  + The permitted temperature range
* In case of wires respectively cores
  + The minimum bend radius
  + The permitted voltage range
  + The insulation material and thickness
* In case of terminals
  + The gender
  + The pull-out force
  + The permitted current range information in dependency of nominal voltage and core cross section area

### Parts classification

The VEC data format specification must have a dedicated parts classification concept which allows a parts type specific description of technical properties. Unlike the KBL the VEC data format specification shall allow parts to be member of more than just one parts classification.

Afterwards, some examples of parts are listed that must be considered by the VEC parts classification concept.

* Connector housings: female / male and mixed housings, pin connector sockets
* Terminals: female and male pluggable terminals, ring terminals, battery terminals, coaxial terminals, contact bridges, bifurcated contacts, optical fibre contact, end-to-end connectors (core separating), Insulation displacement connectors (non-core separating), comb connector
* Wires: single core wires, multi core wires (regarding possible twisting and shielding)
* Wire protections: tapes, sleeves and tubes, fittings
* Seals: cavity seals, cavity plugs, and sealing mats
* Fixings: clips and cable ties
* Grommets: bend protection sleeves, rubber grommets
* E/E-Components: ECUs, sensors, relays, antennas, batteries, fuses, and component boxes

### Description of OEM-parts interrelated with its components supplier(s) and manufacturer(s)

The VEC data format specification must allow the description of OEM parts interrelated with its components supplier(s) and manufacturer(s). For this, it must be possible to describe a mapping of the relevant part numbers. However, for some special cases, the VEC data format specification must allow the definition of mappings that consider further details as well.

## Additional requirements out of the geometry/topology perspective

### Topology

The VEC data format specification must have a concept to describe topology. This concept must consider

* Topology nodes (without coordinates)
* Directed topology segments which are related to the topology nodes
* Further data like segment length and segment cross section area

The topology concept must be designed as the interrelating element between 2D and 3D views.

The topology concept must enable a stand-alone description of topology information as well as a description in combination with relating part placement information and geometry views (2D and/or 3D).

### Partitioning and mapping of topology

The VEC data format specification must have a partitioning respectively grouping concept for topology.

Furthermore, the topology concept must have a mechanism to describe mappings respectively topology embeddings. This can be relevant e.g. for a part with an own topology specification being integrated into the topology of a wiring harness.

A harness definition including topology might be used twice in the vehicle (e.g. left and right rear door). The topology concept must have a mechanism to express that the topology of both wiring harness instances obey the same product definition.

### Placement

The VEC data format specification must have a placement concept which allows the description where part instances or part usages (a kind of placeholder for a concrete part instance respectively a set of concrete part instances representing configuration dependent alternatives) are located on the topology. The placement concept must consider placements that are either defined by one or more reference points (e.g. for the placement of a cable channel) or a dedicated route (e.g. for the placement definition of a winding). In case of overlaps the required layering must be able to be specified.

### Hierarchical Structuring

The VEC data format specification must have a concept which allows the definition of hierarchically structured segments. There are areas of application where the inner structure of a segment matters (e.g. some wires are taped together with another group of wires that are placed in a tube and around all of them a fixing is placed).

### Topology zones

The VEC data format specification must have a concept to assign topology nodes and segments to specific zones that can be associated with properties or requirements (e.g. hot or wet areas, crash zones). It is possible that segments are not completely in or out of zones.

### Dimensioning and tolerance specification

The VEC data format specification must have a concept which allows explicit dimension definitions between in each case two dedicated locations on the topology (specifying the length of the centre line). For this, it must be possible to address the following elements as dimension anchor points:

* Topology nodes
* The reference points that are used for the placement specification of part instance or a part usage. This includes the start and end points of routes   
  (e.g. taping routes) as well.

In case of dimensions between two anchor points which allow ambiguous route interpretations (because of two or more possible paths) the required path must be able to be specified.

Each dimension definition must offer the possibility to specify a nominal value as well as an upper and lower boundary as tolerance specification.

### General requirements concerning geometry views

The VEC data format specification must have a geometry view concept. The same topology must be able to be described in several geometric views. This includes

* 3D views (which normally show the target-installation of the wiring harness)
* 2D views (which normally present a dimensional drawing)

The geometry view concept must allow each topology relevant view element to unambiguously relate to its corresponding topology counterpart element.

Analogously for topology, the VEC geometry view concept must include a partitioning concept. This means the description of a geometry view must be able to be separated in several partitions. At the same time, it must be possible for each of these partitions to be referred by several geometry views. An example use case for the latter could be one 2D drawing for a right-hand-drive vehicle and another 2D drawing for a left-hand-drive car referring (which means sharing) the same partition for the elements that belong to the rear.

### 3D view

The VEC geometry view concept must enable the description of 3D views of wiring harnesses without further external documents (e.g. 3D-PDF). The following information must be able to be described:

* 3D geometry nodes together with their 3D-coordinates (e.g. the coordinates addressing the target-installation of the wiring harness)
* 3D geometry segments each referring their start- and end 3D geometry node, the tangent vectors at these nodes and the definition of the centre line (BSpline). Further details (segment length and cross section area) must be able to be unambiguously assigned in combination with a related topology description.
* Position and orientation of part instances or part usages (connector housings, fixings, …). Note: As the required placement concept as specified in chapter 3.5.3 is demanded to be based on the coordinates-less topology information it can only describe axial information which means placements in relation to the centre line. This means the placement concept will be neither capable to express radial orientation information of placed parts nor translations between bundle position point and the origin of the placed element.

### 2D view

The VEC geometry view concept must enable the description of basic 2D views of wiring harnesses without further external documents (e.g. SVG). The following information must be able to be described:

* 2D geometry nodes together with their 2D-coordinates and optionally a sheet reference
* 2D geometry segments each referring their start- and end 2D geometry node and giving a rough definition of the connecting line
* Position and orientation of part instances or part usages representing view items

### Extra requirements addressing variance information

Out of the GEO-perspective, at least the following elements must be able to express variance dependency by referring to a certain variant configuration term

* Topology and geometry nodes and segments
* Part instances and part usages
* Placement and dimension specifications
* In case of a geometry view divided into partitions each of those partitions

## Additional requirements out of the connectivity perspective

### Abstraction layers

The VEC data format specification must have a concept for the following three abstraction layers

* Architecture
* Schematics
* Wiring

Concerning the data related to one of these abstraction layers, the concept must enable the description both stand-alone and together with the existing interdependencies.

### Architecture

The following information must be able to be described:

* Network nodes: representatives for actors in the electric system, e.g. actuators, sensors and ECUs
* Network ports: representatives for the pins of a network node
* Nets: representatives of connectivity between the related network ports which means without further electric consumer or transformer in between. Nets shall not make any assumptions about the physical realization of the connection and about the topology. On the basis of nets, it must be possible to define net groups.

### Schematics

The following information must be able to be described:

* Component nodes: representatives for elements in the electric system, e.g. actuators, sensors, ECUs and in comparison, with network nodes additionally inliners and splices. Furthermore, there must be a concept for the rough description of the internal architecture of a component node.
* Component ports: representatives for the pins of the component node
* Connections: representative of a connectivity between the related component ports out of the perspective of a single core wire connection. Connections shall not make any assumptions about the topological representation. On the basis of connections, it must be possible to define connection groups and routing definitions (see chapter 3.3.7).

### Wiring

With regard to the development process, the wiring is the immediate predecessor of the wiring harness definition. With regard to content, the wiring does normally not yet consider geometry aspects.

Behind this background, the VEC wiring concept must enable the definition of all necessary requirements that enable together with a corresponding geometry definition the determination of the concrete part instances (in terms of the KBL “occurrences”). Afterwards, some examples are listed the VEC wiring concept must consider at least. However, which pieces of information are regarded as necessary requirements in detail are normally process specific and the VEC wiring concept must be flexible enough to cope with that.

* EE components: e.g. existing housing and pin components
* Connector housings: e.g. number of slots and cavities
* Terminals: e.g. special pin properties, definition of co-axial terminals
* Wires: e.g. core cross section area, insulation thickness, colour
* Contacting: e.g. the type of contacting, see chapter 3.6.5
* Cavity mounting: the assignment of terminals to cavities, see chapter 3.6.6
* Mating: the relationship between pin and contrary pin, see chapter 3.6.7

### Contacting

The VEC data format specification must have a concept that considers the following types of contacting

* Simple contact: assignment of a terminal (wire reception) to one wire end, in case of pluggable terminals mostly in combination with a cavity mounting definition (see chapter 3.6.6)
* Multi contact: assignment of a terminal (wire reception) to more than one wire ends, in case of pluggable terminals mostly in combination with a cavity mounting definition (see chapter 3.6.6)
* Open wire end: definition of no contacting
* Inliner: a special disconnection point between two or more wiring harnesses.
* IDC (insulation displacement connector): connection between two or more EE components that is realised by one physical wire
* Splice: a connection of two or more wire ends without connector housing or EE component

### Cavity Mounting

The VEC data format specification must have a cavity mounting concept in order to describe

* Simple mountings: assignment of a terminal (terminal reception) to one cavity
* Contact bridge: Assignment of a terminal (terminal reception) to more than one cavity
* Plug-in bridge: Assignment of a terminal (terminal reception) to more than one cavity that is itself not connected to a wire.
* Line bridge: Two terminals that are each assigned to one cavity and at the same time are interconnected by a short wire.
* No cavity mounting: e.g. in case of a pluggable terminal that is directly connected with its counterpart terminal (see chapter 3.6.7).

### Mating

The VEC data format specification must have a mating concept in order to describe the (normally pluggable) connectivity between

* The pins/terminals (in case of coax-terminals and piggyback terminals even the relevant terminal receptions) of an EE component and the contrary pins/terminals of the harness connector
* The pins/terminals (in case of a coax-terminal even the relevant terminal receptions) within the two connector housings of an inliner
* Bolts and ring terminals

### Extra requirements addressing variance information

Out of the ELOG-perspective, at least the following elements must be able to express variance dependency by referring to a certain variant configuration term

* All elements of the abstraction layer architecture
* All elements of the abstraction layer schematics
* All elements of the abstraction layer wiring including usages of EE components, connector housings, terminals and wires as well as contacting-, cavity mounting- and mating-definitions.

### Pinning Information

The VEC must provide the possibility to describe electrical interface of EE components. This includes:

* Signals supported by a pin
* Voltage and current values of a Pin for different types and times.
* Variant dependant behaviour of the pin due to the software deployed on an ECU.

## Cross-sectional requirements

### Process specific information

The VEC data format specification must have a concept which describes how to enrich a VEC compliant file with process specific data. For this, each VEC element is requested to be capable to be extended with user-defined attributes.

### Multi-language support for descriptions

For attributes, that carry non-identifying informal text (normally descriptions), the VEC data format specification must have a concept for multi-language support. The language code must be compliant to ISO-639.

### Free choice of measurement units for dimensional quantities

For attributes, that carry dimensional quantities, the VEC data format specification must permit free choice of measurement units. In order to ease the data import, the VEC data format specification must have a special concept for SI data units (like metre – m) and scaled units (like millimetre – mm) as well as combined units (like kg per metre – kg/m).

## Orthogonal grouping concept

The VEC supports information objects from different domains within the development process (e.g. system schematic, component data, wiring harness data, geometry). To support general traceability use cases the VEC shall provide a concept to assign information objects from different domains to groups (e.g. all elements that relate to a specific customer function, a safety relevant requirement).

This concept shall be independent (orthogonal to) of the domain-oriented structure of the model.

## Application Constraints

The VEC data format specification must have concept to apply constraints on the application / usage of specific elements defined in the VEC. This concept must be based on instances (e.g. component nodes in a system schematic or occurrences of a connector in a wiring harness). It is required to define the scope of validity of a certain design / construction and to be complementary to the requirement mentioned in chapter 3.1.2. However, the information required to define the constraints are the same.

## Other / technical requirements

### Data format

The VEC data format is required to be an XML based format. Behind this background, the VEC data format specification must include an XML schema definition. The VEC.XSD must define UNICODE-coding in order to enable special characters to be exchanged.

### Binding the VEC with external models and formats

There are many use cases in which the information represented by a VEC is related to elements outside of the VEC. This relationship to external elements is in many cases relevant information for downstream participants in the process and should be preserved. One obvious approach to achieve this would be to integrate this information into the VEC Model. However, this approach is not satisfying in many cases, because the respective information might not be within the central scope of the VEC or other established standards exist for the representation of this kind of information. Examples for such information are:

* Visual representations like:
  + 3D-Models
  + Wiring Diagrams
  + Component-Symbols
  + Drawings
* Requirements

In order to be able to use these external formats together with the VEC a solution is necessary to create a link / mapping between information in the VEC and information in other external formats.

To make this requirement more clearly, it is explained in the following with a small example:

The Part Master Data of a connector can be described in VEC in a structured way (Cavities and their technical properties, segment connection points, etc.). However, there are many use cases where a visual representation of the connector is needed (e.g. Harness Drawing, Wiring Diagram). This visual representation is often distributed as an SVG-Symbol. If there is no link between the information in the VEC and the visual representation, the information can only be used in a limited way. For more advanced use cases it is necessary to know which element in the VEC (e.g. a Cavity) is represented by which graphical element in the SVG (e.g. a square).

In detail the following requirements must be satisfied:

1. It must be possible to create a mapping between any kind of information in the VEC and elements in an external data source.
2. If the VEC is mapped with some external information, then it must be still possible to read both formats independently (without using the link). E.g. if VEC is linked with an SVG, it must be still possible to read the VEC, without interpreting the SVG and it must be still possible to read SVG with an appropriate viewer, without knowing about the VEC.
3. The used mapping mechanism must not require any structural information about the described data outside of the VEC. Master and source of this kind of data is the VEC. This means for example if a VEC should be linked to SVG it must not be necessary that the SVG complies with a defined structure and has to use certain elements for the representation of its information.
4. The mapping mechanism must be usable for different types of formats. However not all formats are suitable to be used for a mapping. At least they must be structured in some way and elements must be identifiable (e.g. JPEG is not appropriate for such mapping mechanism).
5. The mapping must be standardized, in order to allow an exchange of the mapping information between different systems.

### Tracking of changes to the digital representation of documents

The VEC data format specification must provide a concept to detect / track changes to digital representation of a document. There are cases where the digital content of a document within the VEC changes, even if the PDM relevant information remains unchanged (e.g. the document itself has not been republished, but changes to the exporting system resulted in content changes).

### Extendable Enumerations

The data model of the VEC must support extendable and fixed enumerations (open & closed). For closed enumerations all possible literals are known at the time of the definition of the VEC. For open enumerations some literals are known (which should be standardized), but new literals might emerge in the future. This is often the case, when the literals relate to technical aspects. With new innovations in technology, new literals might be required and shall not require a new version of the VEC schema.

In detail the following requirements must be satisfied:

1. Closed lists of enumeration values have to be supported.
2. Open lists of enumeration values have to be supported.
3. The VEC UML model shall be the single source for documenting open and closed enumerations.
4. The concept shall be supported in the XML schema, too.
5. It shall be possible to validate closed list values against the XML schema
6. It shall be possible to validate the pre-defined open list values against a specific XML schema

# General Guidelines

The most of definition of the VEC is contained in chapter 5 as a detailed model description. However, there are general concepts and guidelines that apply universally and are not limited to an individual model element. Therefore, they are not formulated as part of the model description. These guidelines are defined in the below sections and shall be followed for all implementations of the VEC.

## Handling of Identifiers

The VEC and its XML Schema offer different concepts for the identification of model elements addressing certain requirements and those shall be used accordingly.

### Id Attributes

All *xs:complexType* define an id-Attribute with the type *xs:ID*. These are technical ids that are necessary for the referencing mechanism of the VEC within a single XML file. The semantics, constraints and requirements are defined by the XML Standard and XML Schema itself. These ids do not have any significance outside a VEC file.

### Identification-Elements

Many types defined by the VEC have an “Identification” sub element (E.g. the *PartOccurrence*). This is meant to be a semantic identifier of the object represented by the VEC element. The following rules apply to those identifiers:

1. The expectations defined in the documentation of the VEC model of the corresponding attribute shall be ensured.
2. The identifications shall be unique for a certain element type, at least within its context element. In other words, the VEC model and its representation as XML Schema is a hierarchical data model. That means, that an identification shall be at least unique within its direct parent element (e.g. the identification of a *HousingComponent* shall be unique within its *EEComponentSpecification.*
3. Two elements of different types can have the same *Identification*. However, this is only recommended, when the two VEC elements represent the same domain entity from different points of view, otherwise this shall be avoided as far as possible.
4. In general, it is recommended to keep the *Identifications* stable over the time. This means, that if an object is exported multiple times the *Identification* of it should be the same. However, this is not possible in all cases, for all processes and all tools. Therefore, a process and / or tool creating VEC files should describe for all elements, under which conditions *Identifications* are stable or new ones are created.

### AliasIdentifications

Certain elements have the possibility to define AliasIdentifications in addition to their unique identifications. These are identifiers of the object in a different scope, system or process. One use case of this kind of ids is the creation of traceability links.

Examples for usages of the AliasIdentification are:

* The identifier of a connector in the electrological process (with geometric variants)
* The identifier of a node or segment in a MCAD tool
* An assigned UUID of an element.

## Extension Mechanisms

If the well-defined data structures and fields are not sufficient for the specific needs of a process or a tool, the VEC provides powerful extension mechanisms. Namely the extension mechanisms are custom properties and open enumerations (see the corresponding chapters in the model description).

However, it should be considered that information transported via these mechanisms is not standardized and is always subject to an individual agreement between interface partners. Therefore, these mechanisms shall be used with extreme caution.

It is strictly forbidden to use these mechanisms for the transfer of information that is already standardized within the VEC. In particular it is not permitted:

* To store information in custom properties where already well-defined concepts exist in the VEC to store the same information, e.g. using a custom property instead of an attribute or a more specific class in inheritance tree.
* To use self-defined OpenEnumeration-literals when well-defined literals with the same semantics already exist.

VEC-Files that do not obey to these rules are noncompliant to this data format specification.

If the extension mechanisms are used, it shall always be considered if these extensions might be a valid feature request for the VEC Standard.

## Type Inheritance

The VEC uses an object-oriented class and inheritance concept. The following clarifications apply to its use:

* Only non-abstract classes can be instantiated.
* In an inheritance hierarchy, the choice of the used type represents a semantic information itself. For example, the usage of a PluggableTerminalSpecification is a more specific information than the usage of a TerminalSpecification. It is not required to use the more specific class if the information is not available or it should not be transmitted. However, it is not permitted to use the more general class and transfer the information of the more specific class in a custom property, or similar (e.g. use the TerminalSpecification with a custom property “type=pluggable”).

## Default- and Missing-Value Handling

For various reasons, there may be attributes of entities where no value can be exported, or a special semantics is required. The cases are:

* The information is not supported by the system / process. So it is never available for this system / process.
* The information is supported by the system; however, the value is not defined by the user.
* The information is explicitly defined as “arbitrary” for the use case (e.g. the part version in a bill of material or a compatibility statement).

All cases might exist for mandatory attributes as well as for optional attributes. Due to the design, numerical values in the VEC and its high level of optionality the following definition of special values should be only relevant for *string*-Attributes:

|  |  |  |
| --- | --- | --- |
|  | Mandatory Attribute | Optional Attribute |
| Unsupported | <tag>/NULL</tag> | omitted tag |
| Undefined | <tag></tag> | <tag></tag> |
| Arbitrary | <tag>/ANY</tag> | <tag>/ANY</tag> |

* **“/NULL” & “/ANY”** means, that the attributes with the name “tag” in the VEC receive these values.
* **<tag></tag>** means, that an attribute with the name “tag” and an undefined value is represented in the VEC as an existing XML element with no value (no contained text() node).
* **Omitted tag**: means the element tag for the attribute is not present in the VEC.

## Instantiation of Model Structures

There are various locations in the VEC model where structures / patterns are defined and used / instantiated somewhere else (e.g. a connector with its slots and cavities). In most cases, the elements in the definition of a structure have corresponding elements in the instancing (e.g. ConnectorHousingSpecification 🡪 ConnectorHousingRole, Slot 🡪 SlotReference & Cavity 🡪 CavityReference).

In cases where defined structures are instantiated, these structures shall be instantiated completely. That means, for every element in the structural definition a corresponding element in the instancing shall exist, regardless if it is used in the respective VEC or not (e.g. for each Cavity of a ConnectorHousingSpecification, a CavityReference in the corresponding ConnectorHousingRole shall exist). This applies to the following list of structures, which is here for reasons of clarification and which is not exhaustive:

* Connectors
* Wires
* EEComponents
* CompositeParts (e.g. Assemblies or Modules)

## VEC-Package

### Background

A Vehicle Electric Container (VEC) is a single XML file following the structure defined in the VEC XML schema. It contains all the information of a harness, a set of harnesses, or other related information defined in the VEC specification. A VEC Container can reference other files via the DocumentVersion element and information contained in other files via the ExternalMapping concept.

There are use cases where one wants to exchange the VEC together with these referenced files. There is also the need to exchange a set of VEC files together. The VEC-Package addresses these use cases and specifies the mechanism to exchange VEC files and any associated files as a package.

### Detailed Solution

A VEC-Package is an archive containing at least 2 files:

* One index file (a VEC file)
* One data file (not required to be a VEC file)

Depending on the individual requirements the technical format of the archive can be:

* TAR
* ZIP
* or a zipped tar.

In addition, the archive can contain any number of further data files. There are no restrictions on the type or format of these files. A VEC-Package may contain further VEC files. Or it may contain drawings as SVG, CAD models of the harness or of connectors as JT models, for example.

The structure of the archive is not restricted. A VEC-Package may contain a flat set of files but may also have a folder structure. It is recommended to use a folder structure to organize the files in the archive: e.g. grouping of all drawings, project specific groupings.

There is no naming convention for files and folders inside the VEC-Package defined. It is up to the user to name a folder or a file. However, it is recommended to use the known and established file name extensions for the files in the package. I.e., “.vec” for a VEC file, “.svg” for a SVG file, or “.jt” for a JT file.

A VEC-Package shall contain an index file providing further information about the context of the package. The index file has the reserved name “index.vec”. As the file name suffix already suggests, the index file is a valid VEC file, conforming to the VEC XML schema.

The elements of the index VEC file are restricted to the classes DocumentVersion and PartVersion. The index file contains a DocumentVersion for each file in the package. The attributes of the DocumentVersion are used to provide further information on the files:

* dataFormat: the format of the file in the VEC-Package (as MIME-Type if available).
* documentNumber: the number of the document
* documentVersion: the version of the document
* fileName: the name of the file as it appears in the package, including the folder structure

A DocumentVersion may reference one or more PartVersion objects via *referencedPart* to give further details on the usage of the file. For example, the fact, that a SVG file which represents the wiring diagram of a harness, can be expressed in the index file by a DocumentVersion pointing to a PartVersion, which represents the harness.

# VEC Model Description and XML Representation

# Appendix A: Glossary

ECAD electronic computer aided design

ECU electronic control unit

ELOG A data format specification for the description of electrological data in the wiring harness context.

GEO A data format specification for the description of geometrical data in the wiring harness context.

IDC insulation displacement connector

Item In the context of this recommendation an item is either a part or a document.

KBL harness description list (“Kabelbaumliste”). A data format specification for the description of wiring harness data.

KOMP A data format specification for the description of components data in the wiring harness context.

prostep ivip an international association that has committed itself to developing innovative approaches to solving problems and modern standards for product data management and virtual product creation.

XML Extensible Markup Language

XSD XML schema definition

VDA German association of the automotive industry   
(“Verband der Automobilindustrie”)

VEC vehicle electric container. A data format specification for the description of harness design data cross process steps and supporting tools.

# Appendix B: Data Model Description