| Document Title | Autosar Model Constraints |
|---|---|
| **Document Owner** | AUTOSAR |
| **Document Responsibility** | AUTOSAR |
| **Document Identification No** | 635 |

| | |
|---|---|
| **Document Status** | Final |
| **Part of AUTOSAR Standard** | Classic Platform |
| **Part of Standard Release** | 4.3.1 |

**Document Change History**

| Date | Release | Changed by | Description |
|---|---|---|---|
| 2017-12-08 | 4.3.1 | AUTOSAR Release Management | minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation |
| 2016-11-30 | 4.3.0 | AUTOSAR Release Management | minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation |
| 2015-07-31 | 4.2.2 | AUTOSAR Release Management | minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation |
| 2014-10-31 | 4.2.1 | AUTOSAR Release Management | Editorial changes |
| 2013-10-31 | 4.1.2 | AUTOSAR Release Management | Updated constraints according to changes in SWS and TPS documents |
| 2013-03-15 | 4.1.1 | AUTOSAR Administration | Initial Release |

**Disclaimer**

# Table of Contents

# References

[1] Unified diagnostic services (UDS) – Part 1: Specification and requirements (Release 2006-12)
http://www.iso.org

[2] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

[3] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[4] Specification of RTE Software
AUTOSAR_SWS_RTE

[5] Road vehicles – End-of-life activation of on-board pyrotechnic devices – Part 2: Communication requirements
http://www.iso.org

[6] Information technology – Universal Coded Character Set (UCS)
http://www.iso.org

[7] ISO 17356-4: Road vehicles – Open interface for embedded automotive applications – Part 4: OSEK/VDX Communication (COM)

[8] ISO 17356-3: Road vehicles – Open interface for embedded automotive applications – Part 3: OSEK/VDX Operating System (OS)

[9] Collection of blueprints for AUTOSAR M1 models
AUTOSAR_MOD_GeneralBlueprints

[10] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[11] Specifications of Safety Extensions
AUTOSAR_TPS_SafetyExtensions

[12] XML Path language (XPath)
http://www.w3.org/TR/xpath/

[13] Specification of COM Based Transformer
AUTOSAR_SWS_COMBasedTransformer

[14] SAE J1939-21 Data Link Layer

# 1 Document Information and Content

This auxiliary document provides a collection of constraints for AUTOSAR models. All constraints are copied from template specification and software specification documents, so this document does not introduce any new constraints.

A list of the documents that the constraints originate from can be found in the table of contents. Chapter 2 contains the collected constraints, grouped by source documents. All constraints from the same source document are contained within a single section.

# 2 Autosar Model Constraints

## 2.1 ASWS_TransformerGeneral

**[SWS_Xfrm_CONSTR_09094]** ⌈ If there exists a `XfrmImplementationMapping` which references an `ISignal` or `ISignalGroup` *sig1* and contains the optional parameter `XfrmVariableDataPrototypeInstanceRef` , all `XfrmImplementationMapping` s which reference the same `ISignal` or `ISignalGroup` *sig1* shall contain a `XfrmVariableDataPrototypeInstanceRef` .

⌋*(SRS_Xfrm_00001)*

**[SWS_Xfrm_CONSTR_09095]** ⌈ The `XfrmVariableDataPrototypeInstanceRef` shall refer to the instance of a `VariableDataPrototype` which belongs to a subclass of an `AtomicSwComponentType` .

⌋*(SRS_Xfrm_00001)*

**[SWS_Xfrm_CONSTR_09096]** ⌈ If no `XfrmSignal` exists and hence no `ISignal` or `ISignalGroup` is referenced, `XfrmVariableDataPrototypeInstanceRef` shall be used to reference the instance of the `VariableDataPrototype` which data shall be transformed.

⌋*(SRS_Xfrm_00001)*

## 2.2 SWS_BSWModeManager

**[constr_SWS_BswM_CONSTR_00001]** ⌈ The BswM shall reject configurations where a BswMActionList contains BswMActionListItems with same-valued BswMActionList ItemIndexes.

⌋*()*

**[constr_SWS_BswM_CONSTR_00002]** ⌈ The value of CompuMethod.category referenced by the foreign reference of BswMCompuMethodRef shall be TEXTTABLE.

⌋()

**[constr_SWS_BswM_CONSTR_00003]** ⌈ The BswM shall reject configurations where a BswMDeadlineMonitoringControl container has a BswMDisabledDMPduGroupRef and a BswMEnabledDMPduGroupRef which reference the same PDU Group.

⌋()

**[constr_SWS_BswM_CONSTR_00004]** ⌈ The BswM shall reject configurations where a BswMPduGroupSwitch container has a BswMDisabledPduGroupRef and a Bsw MEnabledPduGroupRef which reference the same PDU Group.

⌋()

## 2.3 SWS_COMManager

**[constr_SWS_ComM_CONSTR_00001]** ⌈ ComM channel's that are referenced by a PNC are not allowed to be referenced by any ComMUsers, if the PNC references at least one EthIfSwitchPortGroup (see figure [REF] Use Case 6). A configuration tool shall reject such a configuration as invalid (error). This constraint is only valid for a host ecu that control an Ethernet switch. In all other UseCases ComMChannels can be referenced by a PNC's and ComMUsers.

⌋()

## 2.4 SWS_DiagnosticCommunicationManager

**[SWS_Dcm_CONSTR_6000] Harmonize the naming between interfaces and modes** ⌈ The shortname of `DcmDspSessionRow` shall match names of Dcm_Ses CtrlType and of the mode declarations of DcmDiagnosticSessionControl. The "DCM_" prefix is mandatory for all shortnames.

⌋()

**[SWS_Dcm_CONSTR_6001] Provide standardized names for ISO standardized diagnostic sessions** ⌈ The following values of `DcmDspSessionLevel` which represent ISO defined diagnostic sessions shall be used for the shortname of `DcmDspSessionRow`:
1 DCM_DEFAULT_SESSION
2 DCM_PROGRAMMING_SESSION
3 DCM_EXTENDED_DIAGNOSTIC_SESSION
4 DCM_SAFETY_SYSTEM_DIAGNOSTIC_SESSION

⌋()

**[SWS_Dcm_CONSTR_6002] Existence of size parameter** ⌈ `DcmDspDataByteSize` shall be present if `DcmDspDataType` is set to: UINT8_N, SINT8_N, UINT16_N, SINT16_N, UINT32_N, SINT32_N or UINT8_DYN.

⌋()

**[SWS_Dcm_CONSTR_6008] Define the usage of `DcmDspRoutineParameterSize` parameter** ⌈ `DcmDspRoutineParameterSize` is only required if `DcmDspRoutineSignalType` is set to SINT8_N, SINT16_N, SINT32_N, UINT8_N, UINT16_N, UINT32_N or VARIABLE_LENGTH.

⌋()

**[SWS_Dcm_CONSTR_6011] Only last parameters in `RID` may have a variable length** ⌈ DcmDspRoutineSignalType with VARIABLE_LENGTH is only valid for the last signal.

⌋()

**[SWS_Dcm_CONSTR_6012] Existence of size parameter** ⌈ `DcmDspPidDataByteSize` shall be present if `DcmDspPidDataType` is set to: UINT8_N, SINT8_N, UINT16_N, SINT16_N, UINT32_N or SINT32_N.

⌋()

**[SWS_Dcm_CONSTR_6018]** ⌈ `DcmDspData` elements used in service 0x2E shall not have DcmDspDataUsePorts set to USE_ECU_SIGNAL.

⌋()

**[SWS_Dcm_CONSTR_6020] Definition of allowed `DID` access** ⌈ Any defined range shall only reference via `DcmDspDidRangeInfoRef` . The sub-containers `DcmDspDidControl` and DcmDspDidDefineinDcmDspDidInfo shall not be used] .

⌋()

**[SWS_Dcm_CONSTR_6021] DID ranges cannot be mapped on DDDIDs, because service 0x2C DDDID does not support the range feature. Practically `DcmDspDidRangeIdentifierLowerLimit` and `DcmDspDidRangeIdentifierUpperLimit` should not include DIDs of the range 0xF200 till 0xF3FF.** ⌈ Any defined range shall only reference `DcmDspDidInfo` via `DcmDspDidRangeInfoRef` , having set `DcmDspDidDynamicallyDefined` == False.

⌋()

**[SWS_Dcm_CONSTR_6023] `DcmDspDidRef` shall not reference the same `DID` reference twice** ⌈ `DcmDspDid` container shall not include the same `DcmDspDidRef` parameters more than once.

⌋()

**[SWS_Dcm_CONSTR_6024] UINT8 shall be used as (implementation) data type for Client-Server interface** ⌈ In case `DcmDspDataUsePort` param-

eter is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}CLIENT\T1\textunderscore {}SERVER , USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}CLIENT\T1\textunderscore {}SERVER , USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}CLIENT\T1\textunderscore {}SERVER\T1\textunderscore {}ERROR , DcmDspDataType shall use UINT8_N or UINT8_DYN.

⌋()

**[SWS_Dcm_CONSTR_6025] Reference to DcmDslResponseOnEvent connection** ⌈ Only one DcmDslROEConnectionRef shall reference DcmDslResponseOnEvent connection.

⌋()

**[SWS_Dcm_CONSTR_6026] Usage of variable data length in case of S/R communication, NvRam access or ECU signal access** ⌈ In case DcmDspDataUsePort is set to { USE\T1\textunderscore {}DATA\T1\textunderscore {}SENDER\T1\textunderscore {}RECEIVER , USE\T1\textunderscore {}DATA\T1\textunderscore {}SENDER\T1\textunderscore {}RECEIVER\T1\textunderscore {}AS\T1\textunderscore {}SERVICE , USE\T1\textunderscore {}BLOCK\T1\textunderscore {}ID , USE\T1\textunderscore {}ECU\T1\textunderscore {}SIGNAL }, the usage of variable data length shall be not allowed.

⌋()

**[SWS_Dcm_CONSTR_6027]** ⌈ The application will inform the Dcm by calling Xxx_Set ActiveDiagnostic() about the ActiveDiagnostic status.

⌋()

**[SWS_Dcm_CONSTR_6028]** ⌈ DcmModeCondition shall either have a DcmBswModeRef or a DcmSwcModeRef or a DcmSwcSRDataElementRef as external reference.

⌋()

**[SWS_Dcm_CONSTR_6029]** ⌈ The values DCM_GREATER_THAN, DCM_GREATER_OR_EQUAL, DCM_LESS_OR_EQUAL and DCM_LESS_THAN shall not used with a Mode reference ( DcmBswModeRef or DcmSwcModeRef ) .

⌋()

**[SWS_Dcm_CONSTR_6030]** ⌈ The ReturnControlToEcu functionnality is existing if at least one of the following parameters are activated : DcmDspDidFreezeCurrentState in ECUC_Dcm_00624 : or DcmDspDidResetToDefault in ECUC_Dcm_00623 : or DcmDspDidShortTermAdjustment in ECUC_Dcm_00625 : .

⌋()

**[SWS_Dcm_CONSTR_6031]** ⌈ The DcmDspData .SHORT-NAME and DcmDspPid-Data .SHORT-NAME shall be distinct.

⌋()

**[SWS_Dcm_CONSTR_6035] Restrictions on size parameter for 16 Bit arrays** ⌈ DcmDspDataByteSize shall be a multiple of 2 if the value is greater than 2 and DcmDspDataType is UINT16_N or SINT16_N.

⌋()

**[SWS_Dcm_CONSTR_6036] Restrictions on size parameter for 32 Bit arrays** ⌈ DcmDspDataByteSize shall be a multiple of 4 if the value is greater than 4 and DcmDspDataType is UINT32_N or SINT32_N.

⌋()

**[SWS_Dcm_CONSTR_6037] Restrictions on datatype usage** ⌈ DcmDsp-DataType shall be UINT8_N or UINT8_DYN, in case DcmDspDataUse-Port is equal to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC\T1\textunderscore {}ER-ROR || USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC || USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC.

⌋()

**[SWS_Dcm_CONSTR_6038] Restrictions on datatype usage** ⌈ DcmDsp-DataType shall be UINT8_N, in case DcmDspDataUsePort is equal to USE\T1\textunderscore {}BLOCK\T1\textunderscore {}ID.

⌋()

**[SWS_Dcm_CONSTR_6039] Signals with variable datalength** ⌈ Only the last signal ( DcmDspDidSignal ) of a DID can have variable datalength ( DcmDspDataType is set to UINT8_DYN).

⌋()

**[SWS_Dcm_CONSTR_6040] Restrictions on size parameter for 16 Bit arrays** ⌈ DcmDspPidDataByteSize shall be a multiple of 2 if the value is greater than 2 and DcmDspPIDDataType is UINT16_N or SINT16_N.

⌋()

**[SWS_Dcm_CONSTR_6041] Restrictions on size parameter for 32 Bit arrays** ⌈ DcmDspPidDataByteSize shall be a multiple of 4 if the value is greater than 4 and DcmDspPIDDataType is UINT32_N or SINT32_N.

⌋()

**[SWS_Dcm_CONSTR_6042] UINT8 shall be used as (implementation) data type for Client-Server interface** ⌈ In case DcmDspPidDataUsePort parameter is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}CLIENT\T1\textunderscore {}SERVER , DcmDspPIDDataType shall use UINT8_N.

⌋()

**[SWS_Dcm_CONSTR_6043] Restrictions on datatype usage** ⌈ Dcm DspPIDDataType shall be UINT8_N in case DcmDspPidDataUsePort is equal to USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC .

⌋()

**[SWS_Dcm_CONSTR_6044]** ⌈ Generic connections shall be consistent. This means that the MetaDataItems and the PduLength of all referenced PDUs of a DcmDslConnection ( DcmDslProtocolRxPduRef , DcmDslProtocolTxPduRef , DcmDslPeriodicTxPduRef , DcmDslRoeTxPduRef )are identical.

⌋()

**[SWS_Dcm_CONSTR_6045]** ⌈ In case the responsibility is on provider side ( DcmDspVehInfoNODIProvResp is set to TRUE), only one DcmDspVehInfoData container shall be allowed.

⌋()

**[SWS_Dcm_CONSTR_6046]** ⌈ In case DcmDspVehInfoDataUsePort is set to FALSE and DcmDspVehInfoDataReadFnc is set to either Dem_DcmGetInfoType Value08 or Dem_DcmGetInfoTypeValue0B then DcmDspVehInfoNODIProvResp shall be set to TRUE.

⌋()

**[SWS_Dcm_CONSTR_6047]** ⌈ Id of the Service identifier configured in DcmDsdSidTabServiceId shall be unique within one DcmDsdServiceTable .

⌋()

**[SWS_Dcm_CONSTR_6048] Composite sub elements accessible only by read** ⌈ Composite sub elements can only be referred from Read DID i.e. Write and Control DID are not supported.

⌋()

**[SWS_Dcm_CONSTR_6049] Limitation to one data element** ⌈ In case DcmDspDidControlMask is set to DCM\T1\textunderscore {}CONTROLMASK\T1\textunderscore {}EXTERNAL , or the DcmDspData element used in service 0x2F has DcmDspDataUsePorts set to USE\T1\textunderscore {}DATA\T1\textunderscore {}SENDER\T1\textunderscore {}RECEIVER || USE\T1\textunderscore {}DATA\T1\textunderscore

{}SENDER\T1\textunderscore {}RECEIVER\T1\textunderscore {}AS\T1\textunderscore {}SERVICE , the upper multiplicity DcmDspDidSignal is limited to 1.

⌋()

**[SWS_Dcm_CONSTR_6050]** ⌈ In case DcmDspDidControlMask is set to DCM_CONTROLMASK_EXTERNAL, or the DcmDspData element used in service 0x2F has DcmDspDataUsePorts set to USE\T1\textunderscore {}DATA\T1\textunderscore {}SENDER\T1\textunderscore {}RECEIVER || USE\T1\textunderscore {}DATA\T1\textunderscore {}SENDER\T1\textunderscore {}RECEIVER\T1\textunderscore {}AS\T1\textunderscore {}SERVICE , the parameter DcmDspDidControlMaskSize shall be present with a value greater than zero.

⌋()

**[SWS_Dcm_CONSTR_6051]** ⌈ The configuration parameter DcmDspDidControlMaskSize shall be only present if DcmDspDidControlMask is equal to DCM\T1\textunderscore {}CONTROLMASK\T1\textunderscore {}EXTERNAL or DCM\T1\textunderscore {}CONTROLMASK\T1\textunderscore {}INTERNAL .

⌋()

**[SWS_Dcm_CONSTR_6053]** ⌈ The aggregation of DcmDspTextTableMapping at DcmDspAlternativeDataType is only valid if the category of the CompuMethod of the DataType referenced by DcmDspAlternativeDataType.DcmApplicationDataType has category set to TEXTTABLE or SCALE_LINEAR_AND_TEXTTABLE.

⌋()

**[SWS_Dcm_CONSTR_6054] Existence of DTCStatusMask** ⌈ DcmDspRoeDTCStatusMask shall be present if DcmDspRoeInitialEventStatus is set to DCM\T1\textunderscore {}ROE\T1\textunderscore {}STOPPED .

⌋()

**[SWS_Dcm_CONSTR_6055] Dependency for DcmDslProtocolMaximumResponseSize** ⌈ DcmDslProtocolMaximumResponseSize shall be only present if DcmPagedBufferEnabled is set to TRUE.

⌋()

**[SWS_Dcm_CONSTR_6056] Dependency for DcmDslProtocolTransType** ⌈ DcmDslProtocolTransType shall be only present if the Dcm_ProtocolType is configured to DCM_ROE_ON_CAN or DCM_ROE_ON_FLEXRAY or DCM_ROE_ON_IP .

⌋()

**[SWS_Dcm_CONSTR_6057] Dependency for `DcmDspDataEcuSignal`** ⌈ `DcmDspDataEcuSignal` shall be only present if `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}ECU\T1\textunderscore {}SIGNAL`.

⌋()

**[SWS_Dcm_CONSTR_6058] Dependency for `DcmDspDataEndianness`** ⌈ In case `DcmDspDataEndianness` is not configured, the `DcmDspDataDefaultEndianness` shall be used instead.

⌋()

**[SWS_Dcm_CONSTR_6059] Dependency for `DcmDspDataFreezeCurrentStateFnc`** ⌈ `DcmDspDataFreezeCurrentStateFnc` shall be only present if:

- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC` or
- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC` or
- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC\T1\textunderscore {}ERROR`

⌋()

**[SWS_Dcm_CONSTR_6060] Dependency for `DcmDspDataGetScalingInfoFnc`** ⌈ `DcmDspDataGetScalingInfoFnc` shall be only present if:

- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC` or
- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC` or
- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC\T1\textunderscore {}ERROR`

⌋()

**[SWS_Dcm_CONSTR_6061] Dependency for `DcmDspDataReadDataLengthFnc`** ⌈ `DcmDspDataReadDataLengthFnc` shall be only present if:

- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC` or

- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC or
- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC\T1\textunderscore {}ERROR

⌋()

**[SWS_Dcm_CONSTR_6062] Dependency for DcmDspDataReadFnc** ⌈ DcmDspDataReadFnc shall be only present if:

- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC or
- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC or
- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC\T1\textunderscore {}ERROR

⌋()

**[SWS_Dcm_CONSTR_6063] Dependency for DcmDspDataResetToDefaultFnc** ⌈ DcmDspDataResetToDefaultFnc shall be only present if:

- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC or
- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC or
- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC\T1\textunderscore {}ERROR

⌋()

**[SWS_Dcm_CONSTR_6064] Dependency for DcmDspDidControlMaskSize** ⌈ DcmDspDidControlMaskSize shall be only present if DcmDspDidControlMask is equal to DCM\T1\textunderscore {}CONTROLMASK\T1\textunderscore {}EXTERNAL or DCM\T1\textunderscore {}CONTROLMASK\T1\textunderscore {}INTERNAL.

⌋()

**[SWS_Dcm_CONSTR_6065] Dependency for DcmDspDataReturnControlToEcuFnc** ⌈ DcmDspDataReturnControlToEcuFnc shall be only present if:

- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC or
- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC or
- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC\T1\textunderscore {}ERROR

⌋()

**[SWS_Dcm_CONSTR_6066] Dependency for DcmDspDataShortTermAdjustmentFnc** ⌈ DcmDspDataShortTermAdjustmentFnc shall be only present if:

- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC or
- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC or
- DcmDspDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC\T1\textunderscore {}ERROR

⌋()

**[SWS_Dcm_CONSTR_6067] Dependency for DcmDspDataBlockIdRef** ⌈ DcmDspDataBlockIdRef shall be only present if DcmDspDataUsePort is set to USE\T1\textunderscore {}BLOCK\T1\textunderscore {}ID.

⌋()

**[SWS_Dcm_CONSTR_6068] Dependency for DcmDspPidDataEndianness** ⌈ In case DcmDspPidDataEndianness is not present, the DcmDspDataDefaultEndianness shall be used instead.

⌋()

**[SWS_Dcm_CONSTR_6069] Dependency for DcmDspPidDataReadFnc** ⌈ DcmDspPidDataReadFnc shall be only present if DcmDspPidDataUsePort is set to USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC.

⌋()

**[SWS_Dcm_CONSTR_6070] Dependency for DcmDspDataEndianness** ⌈ In case DcmDspDataEndianness is not present, the DcmDspDataDefaultEndianness shall be used instead.

⌋()

**[SWS_Dcm_CONSTR_6071] Dependency for `DcmDspStartRoutineFnc` , `DcmDspStopRoutineFnc` , `DcmDspRequestRoutineResultsFnc` , `DcmDspStartRoutineConfirmationFnc` , `DcmDspStopRoutineConfirmationFnc`** ⌈ The following configuration parameters shall only be present if `DcmDspRoutineUsePort` is set to FALSE.

- `DcmDspStartRoutineFnc`
- `DcmDspStopRoutineFnc`
- `DcmDspRequestRoutineResultsFnc`
- `DcmDspStartRoutineConfirmationFnc`
- `DcmDspStopRoutineConfirmationFnc`

⌋()

**[SWS_Dcm_CONSTR_6072] Dependency for `DcmDspRoutineSignalEndianness`** ⌈ In case `DcmDspRoutineSignalEndianness` is not present, the `DcmDspDataDefaultEndianness` shall be used instead.

⌋()

**[SWS_Dcm_CONSTR_6073] Dependency for `DcmDspDataWriteFnc`** ⌈ `DcmDspDataWriteFnc` shall be only present if:

- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}SYNCH\T1\textunderscore {}FNC` or
- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC` or
- `DcmDspDataUsePort` is set to `USE\T1\textunderscore {}DATA\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC\T1\textunderscore {}ERROR`

⌋()

**[SWS_Dcm_CONSTR_6074] Dependency for `DcmDspSecurityMaxAttemptCounterReadoutTime`** ⌈ `DcmDspSecurityMaxAttemptCounterReadoutTime` shall be a multiple and at minimum equal to `DcmTaskTime` .

⌋()

**[SWS_Dcm_CONSTR_6075] Dependency for `DcmDspSecurityCompareKeyFnc`** ⌈ `DcmDspSecurityCompareKeyFnc` shall be configured only if `DcmDspSecurityUsePort` is set to `USE\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC` .

⌋()

**[SWS_Dcm_CONSTR_6076] Dependency for `DcmDspSecurityGetAttemptCounterFnc`** ⌈ `DcmDspSecurityGetAttemptCounterFnc` shall be present only if `DcmDspSecurityUsePort` is set to `USE\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC` and `DcmDspSecurityAttemptCounterEnabled` is set to TRUE.

⌋()

**[SWS_Dcm_CONSTR_6077] Dependency for `DcmDspSecurityGetSeedFnc`** ⌈ `DcmDspSecurityGetSeedFnc` shall be present only if `DcmDspSecurityUsePort` is set to `USE\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC` .

⌋()

**[SWS_Dcm_CONSTR_6078] Dependency for `DcmDspSecuritySetAttemptCounterFnc`** ⌈ `DcmDspSecuritySetAttemptCounterFnc` shall be present only if `DcmDspSecurityUsePort` is set to `USE\T1\textunderscore {}ASYNCH\T1\textunderscore {}FNC` and the `DcmDspSecurityAttemptCounterEnabled` set to TRUE.

⌋()

**[SWS_Dcm_CONSTR_6079] Dependency for `DcmSwcSRDataElementValueRef`** ⌈ `DcmSwcSRDataElementValueRef` shall be present only if `DcmSwcSRDataElementRef` is configured.

⌋()

**[SWS_Dcm_CONSTR_6080] `DcmDspEcuResetRow` container configuration** ⌈ One container `DcmDspEcuResetRow` shall be configured for each `DcmDsdSubService` configured for the `UDS` service ECUReset (0x11).

⌋(*SRS_Diag_04098*)

**[SWS_Dcm_CONSTR_6081] Dependency for `DcmDspDidControlMaskBitPosition`** ⌈ The value configured for `DcmDspDidControlMaskBitPosition` shall be lower than `DcmDspDidControlMaskSize` * 8.

⌋()

## 2.5 SWS_DiagnosticEventManager

**[SWS_Dem_CONSTR_06118] Unique DTC values within a single event memory** ⌈ The `DemDtcValue` shall be unique within all `DTCs` referencing the same event memory.

⌋()

**[SWS_Dem_CONSTR_06119] Unique OBD DTC values within an ECU** ⌈ The `DemDtcValue` shall be unique within all `DTCs` referencing the same event memory.

⌋()

**[SWS_Dem_CONSTR_06120] Dependency for DemGeneralCallbackMonitorStatusChangedFnc** ⌈ The DemGeneralCallbackMonitorStatusChangedFnc shall only be present if DemGeneralInterfaceSupport is set to TRUE."

⌋()

**[SWS_Dem_CONSTR_06121] Dependency for DemMaxNumberEventEntryEventBuffer** ⌈ The DemMaxNumberEventEntryEventBuffer shall only be present if DemEnvironmentDataCapture is set to DEM\T1\textunderscore {}CAPTURE\T1\textunderscore {}SYNCHRONOUS\T1\textunderscore {}TO\T1\textunderscore {}REPORTING.

⌋()

**[SWS_Dem_CONSTR_06122] Dependency for DemOccurrenceCounterProcessing** ⌈ The DemOccurrenceCounterProcessing shall only be present if DemEnvironmentDataCapture is set to DEM\T1\textunderscore {}CAPTURE\T1\textunderscore {}SYNCHRONOUS\T1\textunderscore {}TO\T1\textunderscore {}REPORTING.

⌋()

**[SWS_Dem_CONSTR_06123] Dependency for DemOperationCycleStatusStorage** ⌈ The DemOperationCycleStatusStorage shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06124] Dependency for DemPTOSupport** ⌈ DemPTOSupport shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06125] Dependency for DemAgingCycleCounterThreshold** ⌈ DemAgingCycleCounterThreshold shall only be present if DemAgingAllowed is set to TRUE.

⌋()

**[SWS_Dem_CONSTR_06126] Dependency for DemAgingCycleCounterThresholdForTFSLC** ⌈ DemAgingCycleCounterThresholdForTFSLC shall only be present if DemStatusBitHandlingTestFailedSinceLastClear is set to DEM\T1\textunderscore {}STATUS\T1\textunderscore {}BIT\T1\textunderscore {}AG-

ING\T1\textunderscore {}AND\T1\textunderscore {}DISPLACEMENT
.

⌋()

**[SWS_Dem_CONSTR_06127] Dependency for DemMaxNumberFreezeFrame Records** ⌈ DemMaxNumberFreezeFrameRecords shall only be present if DemTypeOfFreezeFrameRecordNumeration is set to DEM\T1\textunderscore {}FF\T1\textunderscore {}RECNUM\T1\textunderscore {}CALCULATED.

⌋()

**[SWS_Dem_CONSTR_06128] Dependency for DemAgingCycleRef** ⌈ DemAgingCycleRef shall only be present if DemAgingAllowed is set to TRUE.

⌋()

**[SWS_Dem_CONSTR_06129] Dependency for DemFreezeFrameRecNumClassRef** ⌈ DemFreezeFrameRecNumClassRef shall only be present if DemTypeOfFreezeFrameRecordNumeration is set to DEM\T1\textunderscore {}FF\T1\textunderscore {}RECNUM\T1\textunderscore {}CONFIGURED.

⌋()

**[SWS_Dem_CONSTR_06130] Dependency for DemReportBehavior** ⌈ DemReportBehavior shall only be present if DemEventKind is set to DEM\T1\textunderscore {}EVENT\T1\textunderscore {}KIND\T1\textunderscore {}SWC.

⌋()

**[SWS_Dem_CONSTR_06131] Dependency for DemOBDGroupingAssociativeEventsRef** ⌈ DemOBDGroupingAssociativeEventsRef shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06132] Dependency for DemOBDCentralized PID21Handling** ⌈ DemOBDCentralizedPID21Handling shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06133] Dependency for DemOBDCentralized PID31Handling** ⌈ DemOBDCentralizedPID31Handling shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore

{}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06134] Dependency for DemOBDCompliancy** ⌈ DemOBDCompliancy shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06135] Dependency for DemOBDEngineType** ⌈ DemOBDEngineType shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06136] Dependency for DemOBDEventDisplacement** ⌈ DemOBDEventDisplacement shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06137] Dependency for DemOBDInputAcceleratorPedalInformation** ⌈ DemOBDInputAcceleratorPedalInformation shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06138] Dependency for DemOBDInputAmbientPressure** ⌈ DemOBDInputAmbientPressure shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06139] Dependency for DemOBDInputAmbientTemperature** ⌈ DemOBDInputAmbientTemperature shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06140] Dependency for DemOBDInputDistanceInformation** ⌈ DemOBDInputDistanceInformation shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06141] Dependency for DemOBDInputEngineSpeed** ⌈ DemOBDInputEngineSpeed shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06142] Dependency for DemOBDInputEngineTemperature** ⌈ DemOBDInputEngineTemperature shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06143] Dependency for DemOBDInputProgrammingEvent** ⌈ DemOBDInputProgrammingEvent shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06144] Dependency for DemOBDInputVehicleSpeed** ⌈ DemOBDInputVehicleSpeed shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06145] Dependency for DemConsiderPtoStatus** ⌈ DemConsiderPtoStatus shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06146] Dependency for DemDtcValue** ⌈ The OBD DTC DemDtcValue shall only be present if DemOBDSupport

is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06147] Dependency for DemEventOBDReadinessGroup** ⌈ DemEventOBDReadinessGroup shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06148] Dependency on container DemRation** ⌈ The container DemRatio shall only be available if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06149] Dependency on container DemDtr** ⌈ The container DemDtr shall only be available if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06150] Dependency on container DemPidClass** ⌈ The container DemPidClass and aggregated sub-container shall only be present if DemOBDSupport is set to DEM\T1\textunderscore {}OBD\T1\textunderscore {}MASTER\T1\textunderscore {}ECU or DEM\T1\textunderscore {}OBD\T1\textunderscore {}PRIMARY\T1\textunderscore {}ECU.

⌋()

**[SWS_Dem_CONSTR_06151] Dependency on DemCounterBasedFdc ThresholdStorageValue** ⌈ The configuration parameter DemCounterBasedFdcThresholdStorageValue shall only be present if DemFreezeFrameRecordTrigger is set to DEM\T1\textunderscore {}TRIGGER\T1\textunderscore {}ON\T1\textunderscore {}FDC\T1\textunderscore {}THRESHOLD or DemExtendedDataRecordTrigger is set to DEM\T1\textunderscore {}TRIGGER\T1\textunderscore {}ON\T1\textunderscore {}FDC\T1\textunderscore {}THRESHOLD or DemEventMemoryEntryStorageTrigger is set to DEM\T1\textunderscore {}TRIGGER\T1\textunderscore {}ON\T1\textunderscore {}FDC\T1\textunderscore {}THRESHOLD.

⌋()

**[SWS_Dem_CONSTR_06152] Dependency on DemDebounceCounterJumpDown Value** ⌈ DemDebounceCounterJumpDownValue shall only be present if DemDebounceCounterJumpDown is set to TRUE.

⌋()

**[SWS_Dem_CONSTR_06153] Dependency on DemDebounceCounterJumpUp Value** ⌈ DemDebounceCounterJumpUpValue shall only be present if DemDebounceCounterJumpUp is set to TRUE.

⌋()

**[SWS_Dem_CONSTR_06154] Dependency on DemDebounceCounterStorage** ⌈ DemDebounceCounterStorage shall only be present if DemOperationCycleStatusStorage is set to TRUE.

⌋()

**[SWS_Dem_CONSTR_06155] Dependency on DemTimeBasedFdcThreshold StorageValue** ⌈ DemTimeBasedFdcThresholdStorageValue shall only be present if DemFreezeFrameRecordTrigger is set to DEM\T1\textunderscore {}TRIGGER\T1\textunderscore {}ON\T1\textunderscore {}FDC\T1\textunderscore {}THRESHOLD or DemExtendedDataRecordTrigger is set to DEM\T1\textunderscore {}TRIGGER\T1\textunderscore {}ON\T1\textunderscore {}FDC\T1\textunderscore {}THRESHOLD or DemEventMemoryEntryStorageTrigger is set to DEM\T1\textunderscore {}TRIGGER\T1\textunderscore {}ON\T1\textunderscore {}FDC\T1\textunderscore {}THRESHOLD.

⌋()

**[SWS_Dem_CONSTR_06156] Dependency on DemFreezeFrameRecordTrigger** ⌈ DemFreezeFrameRecordTrigger shall only be present if DemTypeOfFreezeFrameRecordNumeration is set to DEM\T1\textunderscore {}FF\T1\textunderscore {}RECNUM\T1\textunderscore {}CONFIGURED.

⌋()

**[SWS_Dem_CONSTR_6101]** ⌈ DemExtendedDataRecordTrigger needs to be configured. DemExtendedDataRecordTrigger shall always be configured, except for internal data elements like occurrence counters.

⌋()

**[SWS_Dem_CONSTR_6103]** ⌈ In case the event combination is disabled, it is not allowed to reference from multiple events to the same dtc.

⌋()

**[SWS_Dem_CONSTR_6104] Limitations on DemMemoryDestinationRef** ⌈ If DemMirrorMemory is configured as DemMemoryDestinationRef , another DemMemoryDestinationRef on the same event of either DemPrimaryMemory or De-

mUserDefinedMemory shall be configured as a prerequisite. The same event shall not be configured two destinations if one is not DemMirrorMemory .

⌋()

**[SWS_Dem_CONSTR_6106]** ⌈ Only directed acyclic graph structures are supported for the dependencies of DemComponent .

⌋()

**[SWS_Dem_CONSTR_6107]** ⌈ Events may be assigned to exactly one DemComponent for which the monitoring is testing the error conditions. Multiple events may be assigned to the same component.

⌋()

**[SWS_Dem_CONSTR_6109]** ⌈ The DTC class is only available for ISO 14229-1 [1] DTCs. It is configurable per DTC optionally (refer to DemWWHOBDDTCClass ).

⌋()

**[SWS_Dem_CONSTR_6110]** ⌈ The WWH-OBD DTC priority shall be according table table_3a_WWH_2d_OBD_20_DTC_20_priority .

⌋()

**[SWS_Dem_CONSTR_6111]** ⌈ An OBD related DTC shall have an aging counter threshold of 40.

⌋()

**[SWS_Dem_CONSTR_6112]** ⌈ An OBD related DTC shall have the Warm-Up cycle as aging cycle.

⌋()

**[SWS_Dem_CONSTR_6113] Configuration of the test failed status bit storage** ⌈ For WWH-OBD ECU the DemStatusBitStorageTestFailed shall be set to True.

⌋()

**[SWS_Dem_CONSTR_6114] Limitations on DemMemoryDestinationRef** ⌈ A DTC can only reference the event memories via DemMemoryDestinationRef to the event memories of the same DemEventMemorySet . The scenario that a DTC references event memories via DemMemoryDestinationRef on different DemEventMemorySet is not supported.

⌋()

**[SWS_Dem_CONSTR_6115]** ⌈ The Dem does not support calls of

- Dem_SetEventStatus
- Dem_ResetEventStatus
- Dem_PrestoreFreezeFrame

- Dem_ClearPrestoredFreezeFrame
- Dem_ResetEventDebounceStatus

with an EventId that is referenced by any of the DemMultiEventTriggeringSlaveEventRef in container DemMultiEventTriggering . These events are exclusively used for internal triggering by calling these APIs for the master event ( DemMultiEventTriggeringMasterEventRef ). The behavior of the Dem is undefined if any of those APIs are called in this situation.

⌋(*SRS_Diag_04165*)

**[SWS_Dem_CONSTR_6116] Limited use of monitor status change callbacks to events reported from SW-Cs only** ⌈ If Dem_SetEventAvailable is called from a Cdd or BSW module, the corresponding monitor status changed callback can only be used as C-function, but not via RTE interface.

⌋()

**[SWS_Dem_CONSTR_6117]** ⌈ The aggregation of DemTextTableMapping at DemAlternativeDataType is only valid if the category of the CompuMethod of the DataType referenced by DemApplicationDataType has category set to TEXTTABLE or SCALE_LINEAR_AND_TEXTTABLE.

⌋()

## 2.6 SWS_FunctionInhibitionManager

**[SWS_Fim_CONSTR_0001]** ⌈ For each configured FiMInhibitionConfiguration , at least one of FiMInhSumRef or FiMInhEventRef or FiMInhComponentRef shall be configured.

⌋()

## 2.7 SWS_RTE

**[SWS_Rte_CONSTR_03510] Exclude usage of OS_SPINLOCK in RteExclusiveAreaImplementation** ⌈ The usage of the enumeration literal OS_SPINLOCK for the parameter RteExclusiveAreaImplMechanism shall be excluded if the parameter RteExclusiveAreaImplMechanism is used in the context of the container RteExclusiveAreaImplementation .

⌋()

**[SWS_Rte_CONSTR_03870]** ⌈ In case that RteDevErrorDetectUninit is configured to true, RteDevErrorDetect shall be configured to true.

⌋()

**[SWS_Rte_CONSTR_09000] `Rte_IFeedback` API may only be used by the `RunnableEntity` s that describe its usage** ⌈ The `Rte_IFeedback` API shall only be used by a `RunnableEntity` that either has a `VariableAccess` in the `dataWriteAccess` role referring to the `VariableDataPrototype` or is triggered by a `DataWriteCompletedEvent` referring to the `VariableAccess` which in turn references the `VariableDataPrototype` .

⌋()

**[SWS_Rte_CONSTR_09001] Whole `DataPrototypeGroup` in role `dpgRequiresCoherency` shall be propagated coherently** ⌈ All `RunnableEntity` s in a `RunnableEntityGroup` with `dataWriteAccess` to data belonging to the same `DataPrototypeGroup` in the role `dpgRequiresCoherency` shall

- Be mapped to the same OS Task

  AND shall

- A) either be scheduled in a way that these `RunnableEntity` s can not be interrupted by `RunnableEntity` s with `dataReadAccess` to (more than one) data belonging to the `DataPrototypeGroup` .
- B) or the `RteImplicitCommunication` shall be configured to ensure a coherent propagation ( `RteCoherentAccess` == true) for reading `RunnableEntity` s `RunnableEntity` s with have as well `dataWriteAccess` to data belonging to the `DataPrototypeGroup` are excluded because inside the calculation chain the latest data values are visible .

⌋()

**[SWS_Rte_CONSTR_09002] The whole `DataPrototypeGroup` shall be read stable for the whole `RunnableEntityGroup` in the role `regRequiresStability`** ⌈ .

All `RunnableEntity` s with `dataReadAccess` to data belonging to the same `DataPrototypeGroup` and which are belonging to the same `RunnableEntityGroup` in the role `regRequiresStability` shall

- either be configured in a way that the chain of `RunnableEntity` s with `dataReadAccess` to the data of the `DataPrototypeGroup` can not be interrupted by any of the `RunnableEntity` (s) with `dataWriteAccess` to data of the `DataPrototypeGroup`
- or the `RteImplicitCommunication` shall be configured to ensure stable data values ( `RteCoherentAccess` == true) for reading `RunnableEntity` s belonging to the `RunnableEntityGroup` .

⌋()

**[SWS_Rte_CONSTR_09005] The references `RteSwcTriggerSourceRef` has to be consistent with the `RteSoftwareComponentInstanceRef`** ⌈ The references `RteSwcTriggerSourceRef` has to be consistent with the `RteSoftwareCompo-`

nentInstanceRef . This means the referenced Trigger / InternalTriggeringPoint has to belong to the AtomicSwComponentType which is referenced by the related SwComponentPrototype .

⌋()

**[SWS_Rte_CONSTR_09006] The references RteBswTriggerSourceRef has to be consistent with the RteBswImplementationRef** ⌈ The references RteBswTriggerSourceRef has to be consistent with the RteBswImplementationRef . This means the referenced Trigger / BswInternalTriggeringPoint has to belong to the BswModuleDescription which is referenced by the related BswImplementation .

⌋()

**[SWS_Rte_CONSTR_09007] *issuedTrigger* and *BswTriggerDirectImplementation* are mutually exclusive** ⌈ A *releasedTrigger Trigger* shall not be referenced by both a *issuedTrigger* and a *BswTriggerDirectImplementation* .

⌋()

**[SWS_Rte_CONSTR_09008] The same Trigger in a *trigger sink* must not be connected to multiple *trigger source* s** ⌈ The same Trigger in a *trigger sink* must not be connected to multiple *trigger source* s.

⌋()

**[SWS_Rte_CONSTR_09009] Synchronized Trigger shall not be referenced by more than one type of access method** ⌈ A synchronized Trigger shall only be referenced by either ExternalTriggeringPoint s, issuedTrigger s or BswTriggerDirectImplementation s.

⌋()

**[SWS_Rte_CONSTR_09010] Worst case execution time shall be less than the GCD** ⌈ The RunnableEntity s or BswSchedulableEntity s worst case execution time shall be less than the GCD of all BswSchedulableEntity s and RunnableEntity s period and offset in activation offset context for RunnableEntity s and BswSchedulableEntity s.

⌋()

**[SWS_Rte_CONSTR_09011] NvMBlockDescriptor related to a RAM Block of a NvBlockSwComponentType shall use NvmBlockUseSyncMechanism** ⌈ The NVRAM Block associated to the NvBlockDescriptor s of a NvBlockSwComponentType shall be configured with the NvMBlockUseSyncMechanism feature enabled, and the NvMWriteRamBlockToNvCallback and NvMReadRamBlockFromNvCallback parameters set to the Rte_GetMirror and Rte_SetMirror API of the NvBlockDescriptor .

⌋()

**[SWS_Rte_CONSTR_09012] Category 1 interrupts shall not access the RTE.** ⌈ Category 1 interrupts shall not access the RTE.

⌋()

**[SWS_Rte_CONSTR_09013] Exactly one mode or one mode transition shall be active** ⌈ Whenever any `RunnableEntity` or `BswSchedulableEntity` is running, there shall always be exactly one mode or one mode transition active of each `ModeDeclarationGroupPrototype`.

⌋()

**[SWS_Rte_CONSTR_09014] *ModeSwitchPoint* (s) and *managedModeGroup* (s) are mutually exclusive for synchronized *ModeDeclarationGroupPrototype* s** ⌈ Only one of two synchronized *ModeDeclarationGroupPrototype* s shall mutual exclusively be referenced by *ModeSwitchPoint* (s) or *managedModeGroup* association(s).

⌋()

**[SWS_Rte_CONSTR_09015] `Rte_Write` API may only be used by the runnable that describe its usage** ⌈ The `Rte_Write` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataSendPoint` role

⌋()

**[SWS_Rte_CONSTR_09016] `Rte_Send` API may only be used by the runnable that describes its usage** ⌈ The `Rte_Send` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataSendPoint` role

⌋()

**[SWS_Rte_CONSTR_09017] `Rte_Switch` API may only be used by the runnable that describes its usage** ⌈ The `Rte_Switch` API may only be used by the runnable that contains the corresponding *ModeSwitchPoint*

⌋()

**[SWS_Rte_CONSTR_09018] `Rte_Invalidate` API may only be used by the runnable that describe its usage** ⌈ The `Rte_Invalidate` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataSendPoint` role

⌋()

**[SWS_Rte_CONSTR_09019] `Rte_Feedback` API may only be used by the runnable that describe its usage** ⌈ A blocking `Rte_Feedback` API may only be used by the runnable that contains the corresponding `WaitPoint`

⌋()

**[SWS_Rte_CONSTR_09020] The blocking `Rte_SwitchAck` API may only be used by the runnable that describes its usage.** ⌈ A blocking `Rte_SwitchAck` API must only be used by the runnable that contains the corresponding `WaitPoint`

⌋()

**[SWS_Rte_CONSTR_09021] `Rte_Read` API may only be used by the runnable that describe its usage** ⌈ The `Rte_Read` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataReceivePointByArgument` role

⌋()

**[SWS_Rte_CONSTR_09022] `Rte_DRead` API may only be used by the runnable that describe its usage** ⌈ The `Rte_DRead` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataReceivePointByValue` role

⌋()

**[SWS_Rte_CONSTR_09023] `Rte_Receive` API may only be used by the runnable that describe its usage** ⌈ The `Rte_Receive` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataReceivePointByArgument` role

⌋()

**[SWS_Rte_CONSTR_09024] `Rte_Call` API may only be used by the runnable that describe its usage** ⌈ The `Rte_Call` API may only be used by the runnable that contains the corresponding `ServerCallPoint`

⌋()

**[SWS_Rte_CONSTR_09025] Blocking `Rte_Result` API may only be used by the runnable that describe the `WaitPoint`** ⌈ The blocking `Rte_Result` API may only be used by the runnable that contains the corresponding `WaitPoint`

⌋()

**[SWS_Rte_CONSTR_09026] `Rte_IWriteRef` may not return values written in previous executions** ⌈ The reference returned by `Rte_IWriteRef` shall not be used by the runnables for reading the value previously written.

⌋()

**[SWS_Rte_CONSTR_09027] `Rte_IStatus` API shall only be used by a `RunnableEntity` describing an read access to the related data** ⌈ The `Rte_IStatus` API shall only be used by a `RunnableEntity` that has a `VariableAccess` in the `dataReadAccess` role referring to the `VariableDataPrototype` to which the status belongs.

⌋()

**[SWS_Rte_CONSTR_09028] `Rte_Enter` and `Rte_Exit` API may only be used by runnables describing its usage** ⌈ The `Rte_Enter` and `Rte_Exit` API may only be used by *Runnable Entities* that contain a corresponding *canEnterExclusiveArea* association

⌋()

**[SWS_Rte_CONSTR_09029] Nested call of `Rte_Enter` and `Rte_Exit` is restricted** ⌈ The `Rte_Enter` and `Rte_Exit` API may only be called nested if different exclusive areas are invoked; in this case exclusive areas shall exited in the reverse order they were entered.

⌋()

**[SWS_Rte_CONSTR_09030] `Rte_Mode` API may only be used by the runnable that describe its usage** ⌈ The `Rte_Mode` API may only be used by the runnable that contains the corresponding `ModeAccessPoint`

⌋()

**[SWS_Rte_CONSTR_09031] `Rte_Mode` API may only be used by the runnable that describe its usage** ⌈ The `Rte_Mode` API may only be used by the runnable that contains the corresponding `ModeAccessPoint`

⌋()

**[SWS_Rte_CONSTR_09032] `Rte_Trigger` API may only be used by the runnable that describe its usage** ⌈ The `Rte_Trigger` API may only be used by the runnable that contains the corresponding `ExternalTriggeringPoint` .

⌋()

**[SWS_Rte_CONSTR_09033] `Rte_IrTrigger` API may only be used by the runnable that describe its usage** ⌈ The `Rte_IrTrigger` API may only be used by the runnable that contains the corresponding `InternalTriggeringPoint` .

⌋()

**[SWS_Rte_CONSTR_09034] `Rte_IsUpdated` API may only be used by the runnable that describe the access to the corresponding data** ⌈ The `Rte_IsUpdated` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataReceivePointByArgument` or `dataReceivePointByValue` role.

⌋()

**[SWS_Rte_CONSTR_09035] `Rte_Start` shall be called only once** ⌈ `Rte_Start` shall be called only once by the EcuStateManager from trusted OS context on a core after the basic software modules required by RTE are initialized.

⌋()

**[SWS_Rte_CONSTR_09036] `Rte_Start` API may only be used after call of `SchM_Init`** ⌈ The `Rte_Start` API may only be used after the *Basic Software Scheduler* is initialized (after termination of the `SchM_Init` ).

⌋()

**[SWS_Rte_CONSTR_09037]** **`Rte_Start` API shall be called on every core** ⌈ The `Rte_Start` API shall be called on every core that hosts AUTOSAR software-components of the ECU.

⌋()

**[SWS_Rte_CONSTR_09038]** **`Rte_Stop` shall be called before BSW shutdown** ⌈ `Rte_Stop` shall be called by the EcuStateManager before the basic software modules required by RTE are shut down.

⌋()

**[SWS_Rte_CONSTR_09039]** **`Rte_PartitionTerminated` shall be called only once** ⌈ `Rte_PartitionTerminated` shall be called only once by the Protection Hook.

⌋()

**[SWS_Rte_CONSTR_09040]** **`Rte_PartitionRestarting` shall be called only onc** ⌈ `Rte_PartitionRestarting` shall be called only once by the ProtectionHook.

⌋()

**[SWS_Rte_CONSTR_09041]** **`Rte_RestartPartition` shall be called from `RestartTask`** ⌈ `Rte_RestartPartition` shall be called only in the context of the `RestartTask` of the given partition.

⌋()

**[SWS_Rte_CONSTR_09042]** **`Array Implementation Data Type`s needs at least one element** ⌈ The `arraySize` defining number of elements in one dimension of an *Array Implementation Data Type* shall be an integer that is $\geq 1$ for each dimension.

⌋()

**[SWS_Rte_CONSTR_09043]** **`Structure Implementation Data Type`s needs at least one element** ⌈ A structure shall include at least one element defined by a `ImplementationDataTypeElement` .

⌋()

**[SWS_Rte_CONSTR_09045] The upper two bits of the of the server return value are reserved** ⌈ Only the least significant six bit of the return value of a server runnable shall be used by the application to indicate an error. The upper two bit shall be zero.

⌋()

**[SWS_Rte_CONSTR_09046]** **`SchM_Enter` and `SchM_Exit` API may only be used by `BswModuleEntity` s describing its usage** ⌈ The `SchM_Enter` and `SchM_Exit` API may only be used by `BswModuleEntity` s that contain a corresponding `canEnterExclusiveArea` association

⌋()

**[SWS_Rte_CONSTR_09047] Nested call of `SchM_Enter` and `SchM_Exit` API is restricted** ⌈ The `SchM_Enter` and `SchM_Exit` API may only be called nested if different exclusive areas are invoked; in this case exclusive areas shall exited in the reverse order they were entered.

⌋()

**[SWS_Rte_CONSTR_09048] `SchM_Exit` API may only be used by `BswModuleEntity` s that describe its usage** ⌈ The `SchM_Exit` API may only be used by `BswModuleEntity` s that contain a corresponding `canEnterExclusiveArea` association

⌋()

**[SWS_Rte_CONSTR_09049] `SchM_Switch` API may only be used by `BswModuleEntity` s that describe its usage** ⌈ The `SchM_Switch` API may only be used by `BswModuleEntity` s that contain a corresponding `managedModeGroup` association

⌋()

**[SWS_Rte_CONSTR_09050] `SchM_Mode` API may only be used by `BswModuleEntity` s that describe its usage** ⌈ The `SchM_Mode` API may only be used by `BswModuleEntity` s that contain a corresponding `managedModeGroup` association or `accessedModeGroup` association

⌋()

**[SWS_Rte_CONSTR_09051] `SchM_Mode` API may only be used by `BswModuleEntity` s that describe its usage** ⌈ The `SchM_Mode` API may only be used by `BswModuleEntity` s that contain a corresponding `managedModeGroup` association or `accessedModeGroup` association

⌋()

**[SWS_Rte_CONSTR_09052] `SchM_SwitchAck` API may only be used by `BswModuleEntity` s that describe its usage** ⌈ The `SchM_SwitchAck` API may only be used by `BswModuleEntity` s that contain a corresponding `managedModeGroup` association

⌋()

**[SWS_Rte_CONSTR_09053] `SchM_Trigger` API may only be used by the `BswModuleEntity` s that describe its usage** ⌈ The `SchM_Trigger` API may only be used by the `BswModuleEntity` that contains the corresponding `issuedTrigger` association.

⌋()

**[SWS_Rte_CONSTR_09054] `SchM_ActMainFunction` API may only be used by the `BswModuleEntity` s that describe its usage** ⌈ The `SchM_ActMainFunction`

API may only be used by the `BswModuleEntity` that contains the corresponding `activationPoint` association.

⌋()

**[SWS_Rte_CONSTR_09055] `SchM_Init , SchM_Start , SchM_StartTiming` shall be called only once** ⌈ `SchM_Init` , `SchM_Start` , `SchM_StartTiming` shall be called only once by the *EcuStateManager* on each core after the basic software modules required by the *Basic Software Scheduler* part of the RTE are initialized.

⌋()

**[SWS_Rte_CONSTR_09056] `SchM_Deinit` API may only be used after the was RTE finalized** ⌈ The `SchM_Deinit` API may only be used after the RTE finalized (after termination of the `Rte_Stop` )

⌋()

**[SWS_Rte_CONSTR_09057] `SchM_Deinit` shall be called before shut down of BSW** ⌈ `SchM_Deinit` shall be called by the *EcuStateManager* before the basic software modules required by *Basic Software Scheduler* part are shut down.

⌋()

**[SWS_Rte_CONSTR_09058] `BswSchedulableEntity` is not allowed to have service arguments or return value** ⌈ The Basic Software Scheduler requires that the `BswModuleEntry` has no service arguments (unless |SchM_ActivatingEvent| is enabled) and no return value.

⌋()

**[SWS_Rte_CONSTR_09059] Usage of *Basic Software Scheduler* API prerequisites the include of the *Module Interlink Header File*** ⌈ Each BSW module implementation shall include its *Module Interlink Header File* if it uses *Basic Software Scheduler* API or if it implements `BswSchedulableEntity` s.

⌋()

**[SWS_Rte_CONSTR_09060] `Rte_Init` API may only be used after call of `Rte_Start`** ⌈ The `Rte_Init` API may only be used after the *RTE* is initialized (after termination of the `Rte_Start` ).

⌋()

**[SWS_Rte_CONSTR_09061] `Rte_StartTiming` API may only be used after call of `Rte_Start`** ⌈ The `Rte_StartTiming` API may only be used after the *RTE* is initialized (after termination of the `Rte_Start` ).

⌋()

**[SWS_Rte_CONSTR_09062] Entire mapping of `on-entry Runnable Entities` for `initialMode` to `RteInitializationRunnableBatch` containers** ⌈ Either all or none of the `on-entry Runnable Entities` of a particular `mode machine in-`

stance for the initialMode shall be mapped to RteInitializationRunnable-Batch containers.

⌋()

**[SWS_Rte_CONSTR_09063] Restricted kinds of RTEEvent s which may mapped to RteInitializationRunnableBatch containers** ⌈ Only SwcMod-eSwitchEvent s with activation = onEntry and referring to the initialMode or InitEvent s may be mapped to RteInitializationRunnableBatch containers with the means of a RteUsedInitFnc reference.

⌋()

**[SWS_Rte_CONSTR_09064] A single RteInitializationRunnableBatch container may not handle RTEEvent s of different partitions** ⌈ All RTEEvent s mapped to a RteInitializationRunnableBatch container may only trigger RunnableEntity s belonging to the same partition.

⌋()

**[SWS_Rte_CONSTR_09076] SchM_Result API may only be used by the Bsw ModuleEntity that describe its usage** ⌈ The SchM_Result API may only be used within the BswModuleEntity that references the corresponding BswAsyn-chronousServerCallResultPoint using a callPoint association.

⌋()

**[SWS_Rte_CONSTR_09077] SchM_Send API may only be used by the Bsw ModuleEntity that describes its usage** ⌈ The SchM_Send API may only be used within the BswModuleEntity that references the VariableDataPrototype using a dataSendPoint .

⌋()

**[SWS_Rte_CONSTR_09078] SchM_Receive API may only be used by the Bsw ModuleEntity that describes its usage** ⌈ The SchM_Receive API may only be used within the BswModuleEntity that references the VariableDataPrototype using a dataReceivePoint .

⌋()

**[SWS_Rte_CONSTR_09079] SchM_Call API may only be used by the BswModule Entity that describe its usage** ⌈ The SchM_Call API may only be used within the BswModuleEntity that references the corresponding BswSynchronousServer-CallPoint respectively BswAsynchronousServerCallPoint using a call-Point association.

⌋()

**[SWS_Rte_CONSTR_09080] The *shortNames* of *PortInterfaces* shall be unique within a software component if it supports multiple instantiation or indirec-tAPI attribute is set to 'true'** ⌈ The *shortNames* of *PortInterfaces* shall be unique

within a software component for each set of PPortPrototypes or RPortPrototypes if the software component supports multiple instantiation or if the `indirectAPI` attribute is set to 'true' for at least one require or provide port.

This is required to generate distinguishable Port Data Structure data types.

⌋()

**[SWS_Rte_CONSTR_09081] Mapping to partition vs the value of VariableAccess.scope** ⌈ For every connection between `SwComponentPrototype` s mapped to different partitions the value of `VariableAccess . scope` shall not be set to VariableAccessScopeEnum.communicationIntraPartition.

⌋()

**[SWS_Rte_CONSTR_09082] `RtePositionInTask` and `RteBswPositionInTask` values shall be unique in a particular context** ⌈ `RtePositionInTask` and `RteBswPositionInTask` shall have unique values for any particular task in the case `RTEEvent` s and `BswEvent` s are mapped to `OsTask` s and shall have unique values for any particular scope of direct invocation in the case that the a direct function call is configured. The only exception are `RtePositionInTask` values for `RteEventToTaskMapping` s mapping the `OperationInvokedEvent` s for several `operation` s to the same `server runnable`s .

⌋()

**[SWS_Rte_CONSTR_09083] `Rte_IRead` API may only be used by the runnable that describe its usage** ⌈ The `Rte_IRead` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataReadAccess` role.

⌋()

**[SWS_Rte_CONSTR_09084] `Rte_IWrite` API may only be used by the runnable that describe its usage** ⌈ The `Rte_IWrite` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataWriteAccess` role.

⌋()

**[SWS_Rte_CONSTR_09085] `Rte_IWriteRef` API may only be used by the runnable that describe its usage** ⌈ The `Rte_IWriteRef` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataWriteAccess` role.

⌋()

**[SWS_Rte_CONSTR_09086] `Rte_IInvalidate` API may only be used by the runnable that is describing an write access to the data** ⌈ The `Rte_IInvalidate` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `dataWriteAccess` role to the `VariableDataPrototype` where the associated `InvalidationPolicy` of the `VariableDataPrototype` is set to `keep` or `replace` .

⌋()

**[SWS_Rte_CONSTR_09087] `Rte_IrvIRead` API may only be used by the runnable that describe its usage** ⌈ The `Rte_IrvIRead` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `readLocalVariable` role.

⌋()

**[SWS_Rte_CONSTR_09088] `Rte_IrvIWrite` API may only be used by the runnable that describe its usage** ⌈ The `Rte_IrvIWrite` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `writtenLocalVariable` role.

⌋()

**[SWS_Rte_CONSTR_09089] `Rte_IrvRead` API may only be used by the runnable that describe its usage** ⌈ The `Rte_IrvRead` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `readLocalVariable` role.

⌋()

**[SWS_Rte_CONSTR_09090] `Rte_IrvWrite` API may only be used by the runnable that describe its usage** ⌈ The `Rte_IrvWrite` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `writtenLocalVariable` role.

⌋()

**[SWS_Rte_CONSTR_09091] `RteSwNvRamMappingRef` and `RteSwNvBlockDescriptorRef` are excluding each other** ⌈ If an `RteSwNvBlockDescriptorRef` is defined there shall be no `RteSwNvRamMappingRef` , `RteNvmRomBlockLocationSymbol` and `RteNvmRamBlockLocationSymbol` defined. If an `RteSwNvRamMappingRef` is defined there shall be no `RteSwNvBlockDescriptorRef` defined.

⌋()

**[SWS_Rte_CONSTR_09092] `Rte_IrvIWriteRef` API may only be used by the runnable that describe its usage** ⌈ The `Rte_IrvIWriteRef` API may only be used by the runnable that contains the corresponding `VariableAccess` in the `writtenLocalVariable` role.

⌋()

**[SWS_Rte_CONSTR_09093] `Rte_IrvIWriteRef` may not return values written in previous executions** ⌈ The reference returned by `Rte_IrvIWriteRef` shall not be used by the runnables for reading the value previously written.

⌋()

## 2.8 SWS_SAEJ1939DiagnosticCommunicationManager

**[constr_SWS_J1939Dcm_CONSTR_6201]** ⌈ J1939DcmModeCondition shall have either a J1939DcmBswModeRef or a J1939DcmSwcModeRef or a J1939DcmSwcSRDataElementRef as external reference.

⌋()

**[constr_SWS_J1939Dcm_CONSTR_6202]** ⌈ The values J1939DCM_GREATER_THAN, J1939DCM_GREATER_OR_EQUAL, J1939DCM_LESS_OR_EQUAL and J1939DCM_LESS_THAN shall not be used with a Mode reference (J1939DcmBswModeRef or J1939DcmSwcModeRef).

⌋()

## 2.9 SWS_WatchdogManager

**[constr_SWS_WdgM_CONSTR_6500] Interface provision in MCU driver** ⌈ The parameter WdgMImmediateReset [ECUC_WdgM_00339] may only be set to TRUE if the McuPerformResetApi (defined in SWS_Mcu_Driver) is set to TRUE.

⌋()

**[constr_SWS_WdgM_CONSTR_6501] Only non-trusted OS-Application can be restarted** ⌈ WdgMOsApplicationRef shall not point to a trusted OS-Application (i.e. where OsTrusted the of OsApplication is TRUE).

⌋()

**[constr_SWS_WdgM_CONSTR_6502]** ⌈ A unique Supervised Entity identifier for each Supervised Entity is provided in configuration parameter WdgMSupervisedEntityID (see [ECUC_WdgM_00304]). The Identifier shall be unique in the scope of the Watchdog Manager module.

⌋()

**[constr_SWS_WdgM_CONSTR_6503]** ⌈ Each BSW module shall use its module ID as the Supervised Entity ID.

⌋()

**[constr_SWS_WdgM_CONSTR_6504]** ⌈ No SW-Cs shall have as Supervised Entity ID a value of any BSW Module ID, regardless which BSW Modules are deployed.

⌋()

**[constr_SWS_WdgM_CONSTR_6505]** ⌈ Deadline Supervision (WdgMDeadlineSupervision) of a Supervised Entity shall refer to Checkpoints (WdgMDeadlineStartRef, WdgMDeadlineEndRef) that both belong to that Supervised Entity. In other words, any of the referred Checkpoints shall not belong to other Supervised Entities.

⌋()

**[constr_SWS_WdgM_CONSTR_6506]** ⌈ Internal Transitions (see WdgMInternalTransition) in a Supervised Entity shall not connect Checkpoints that do not both belong to the same Supervised Entity.

⌋()

**[constr_SWS_WdgM_CONSTR_6507]** ⌈ A Checkpoint shall not belong to more than one Internal Graph.

⌋()

**[constr_SWS_WdgM_CONSTR_6508]** ⌈ A Checkpoint shall not belong to an External Graph and to an Internal Graph; this applies across all modes.

⌋()

**[constr_SWS_WdgM_CONSTR_6509]** ⌈ In a given mode, a Checkpoint shall not belong to more than one External Graph.

⌋()

**[constr_SWS_WdgM_CONSTR_6510]** ⌈ The following shall be available for the operation supervision functions of Watchdog Manager:

- availability of initialized Wdg Interface,
- availability of initialized OS,
- initialized WdgM - by invocation of WdgM_Init() function.

⌋()

**[constr_SWS_WdgM_CONSTR_6511]** ⌈ It shall be ensured by the callers of Wdg M module, that the functions WdgM_DeInit, WdgM_Init and WdgM_SetMode are not invoked concurrently to WdgM_MainFunction.

⌋()

**[constr_SWS_WdgM_CONSTR_6512]** ⌈ Any ordered set of two Checkpoints shall not have more than one Deadline Supervision (WdgMDeadlineSupervision) defined.

⌋()

## 2.10 TPS_BSWModuleDescriptionTemplate

**[constr_1275] Applicability of reference `startsOnEvent` for `BswSchedulableEvent`** ⌈ The reference `BswScheduleEvent . startsOnEvent` shall only refer to a `BswSchedulableEntity` .

⌋()

**[constr_1276] Applicability of reference `startsOnEvent` for `BswOperationInvokedEvent`** ⌈ The reference `BswOperationInvokedEvent . startsOnEvent` shall only refer to a `BswCalledEntity` .

⌋()

**[constr_4013] BSW service identifier** ⌈ For Standardized Interfaces, this identifier is defined in the AUTOSAR Software Specification (SWS) of the module. In case the C-function prototype represented by the entry is not standardized, it still can be used optionally, but its value must differ from the standardized ones.

⌋()

**[constr_4014] Call type and execution context** ⌈ Within a given `BswModuleEntry` , the following constraint holds for its attributes:

- `callType ==’interrupt’` is not allowed together with `executionContext ==’task’ or ==’hook’`
- `callType ==’scheduled’` is not allowed together with `executionContext ==’interruptCat1’ or ==’interruptCat2’`
- other combinations of these two enums are allowed

⌋()

**[constr_4015] `calledEntry` constraints for direct calls** ⌈ The following holds if `callPoint` is aggregated as an instance of `BswDirectCallPoint` :

- `BswModuleEntity . callPoint . calledEntry . executionContext` must be identical to `BswModuleEntity . implementedEntry . executionContext`
- `BswModuleEntity . callPoint . calledEntry . callType` must have the value `’regular’` or `’callback’`

⌋()

**[constr_4016] `BswCalledEntity` constraints** ⌈

- `BswCalledEntity . implementedEntry . callType` must be `’regular’` or `’callback’`
- `BswCalledEntity . implementedEntry . executionContext` is in general not restricted, but see constr_4076 for constraints on the server side of a Client-Server communication.

⌋()

**[constr_4017] `BswSchedulableEntity` constraints** ⌈

- `BswModuleEntity . implementedEntry . callType` must be `’scheduled’`

- BswModuleEntity . implementedEntry . executionContext must be 'task'

⌋()

**[constr_4018] BswInterruptEntity constraints** ⌈

- BswInterruptEntity . implementedEntry . callType must be 'interrupt'
- BswInterruptEntity . implementedEntry . executionContext must be 'interruptCat1' if and only if BswInterruptEntity . interruptCategory is 'Cat1'
- BswInterruptEntity . implementedEntry . executionContext must be 'interruptCat2' if and only if BswInterruptEntity . interruptCategory is 'Cat2'

⌋()

**[constr_4019] BSW module identifier** ⌈ BswModuleDescription . moduleId shall refer to the identifier of the standardized AUTOSAR modules according to [2] , if applicable Note that there may be more than one module in an ECU software with the same identifier, e.g. according to the standard Complex Drivers all have the same identifier. . Otherwise (e.g. for ICC2 clusters) the identifier must either be empty or chosen differently from the ones given in [2] .

⌋()

**[constr_4020] Categories of BswModuleDescription** ⌈ Only categories listed in table table_3a_BSWMD_Categories are allowed. Other values or an empty value are not allowed.

⌋()

**[constr_4021] Implementation policy of function pointer target** ⌈
A BswModuleEntry can only be used as target of a function pointer ( SwPointerTargetProps . functionPointerSignature ), if its swServiceImplPolicy is 'standard' .

⌋()

**[constr_4022] BswModuleEntity only uses the module's interface** ⌈

- BswModuleEntity . implementedEntry must refer to an element declared as implementedEntry of the enclosing BswModuleDescription
- BswModuleEntity . callPoint . calledEntry - where callPoint is instantiated from BswDirectCallPoint - must refer to an element declared as expectedEntry or implementedEntry of the enclosing BswModuleDescription .

- BswModuleEntity . callPoint . calledEntry - where callPoint is instantiated from BswSynchronousServerCallPoint or BswAsynchronousServerCallPoint - must refer to an element declared as requiredClientServerEntry of the enclosing BswModuleDescription .
- BswModuleEntity . callPoint - where callPoint is instantiated from BswAsynchronousServerCallResultPoint - must refer to an BswAsynchronousServerCallPoint declared in turn as callPoint of the same BswModuleEntity .
- BswModuleEntity . issuedTrigger must refer to an element declared as releasedTrigger of the enclosing BswModuleDescription
- BswModuleEntity . managedModeGroup must refer to an element declared as providedModeGroup of the enclosing BswModuleDescription
- BswModuleEntity . accessedModeGroup must refer to an element declared as requiredModeGroup of the enclosing BswModuleDescription
- BswModuleEntity . dataSendPoint . accessedVariable must refer to an element declared as providedData of the enclosing BswModuleDescription
- BswModuleEntity . dataReceivePoint . accessedVariable must refer to an element declared as requiredData of the enclosing BswModuleDescription
- an accessedModeGroup should be allowed to refer to an element declared as providedModeGroup

⌋()

**[constr_4023] External trigger must belong to the interface** ⌈ A BswExternalTriggerOccurredEvent must refer to a Trigger that is declared via BswModuleDescription . requiredTrigger for the same module.

⌋()

**[constr_4024] Semantics of BSW mode switch event** ⌈ If BswModeSwitchEvent . activation has the value onTransition BswModeSwitchEvent shall refer to two different modes belonging to the same instance of ModeDeclarationGroup, their order defining the direction of the transition. In all other cases, BswModeSwitchEvent shall refer to exactly one mode.

⌋()

**[constr_4025] Modes used by BSW mode switch event** ⌈ The ModeDeclaration used by BswModeSwitchEvent must belong to the ModeDeclarationGroupPrototype referred as BswInternalBehavior . entity . accessedModeGroup of the enclosing BswInternalBehavior .

⌋()

**[constr_4026] Mode group used by BSW mode switch acknowledge event** ⌈ The `ModeDeclarationGroupPrototype` used by `BswModeSwitchedAckEvent` must be referred as `BswModuleDescription . providedModeGroup` by the same module.

⌋()

**[constr_4028] Semantics of memory section type** ⌈ `sectionType` must be semantically compatible to the usage of the enclosing `SwAddrMethod` , this means especially that if `SwAddrMethod` is associated by `ExecutableEntity` -s, the `sectionType` must be usable as code section, if it is associated by `SwDataDefProps` , `sectionType` must be usable as data section.

⌋()

**[constr_4029] Measured stack usage** ⌈ The attribute values of `MeasuredStackUsage` must fulfill:
`minimumMemoryConsumption <= averageMemoryConsumption <= maximumMemoryConsumption`

⌋()

**[constr_4030] Measured heap usage** ⌈ The attribute values of `MeasuredHeapUsage` must fulfill:
`minimumMemoryConsumption <= averageMemoryConsumption <= maximumMemoryConsumption`

⌋()

**[constr_4031] Analyzed execution time** ⌈ The attribute values of `AnalyzedExecutionTime` must fulfill:
`bestCaseExecutionTime <= bestCaseExecutionTime`

⌋()

**[constr_4032] Measured execution time** ⌈ The attribute values of `MeasuredExecutionTime` must fulfill:
`minimumExecutionTime <= nominalExecutionTime <= maximumExecutionTime`

⌋()

**[constr_4033] Simulated execution time** ⌈ The attribute values of `SimulatedExecutionTime` must fulfill:
`minimumExecutionTime <= nominalExecutionTime <= maximumExecutionTime`

⌋()

**[constr_4034] Target and context of MC emulation reference** ⌈ Within one `ImplementationElementInParameterInstanceRef` , the `target` must refer to a sub-element of the `ParameterDataPrototype` which is referred as `context` .

⌋()

**[constr_4038] `bswModuleDependency` must refer to a different module** ⌈

- `BswModuleDescription . bswModuleDependency . targetModuleId` (if given) must differ from `BswModuleDescription . moduleId` . This does not hold if the value is 254 (used for IO Hardware Abstraction modules) or 255 (used for Complex Driver modules).
- `BswModuleDependency . targetModuleRef` (if given) must differ from the package location of the `BswModuleDescription` that owns the `BswModuleDependency` .

⌋()

**[constr_4039] Semantics of `SwcBswMapping`** ⌈ An `SwcBswMapping` is only valid, if the referred `SwcInternalBehavior` is aggregated by a `ServiceSwComponentType` , `EcuAbstractionSwComponentType` or `ComplexDeviceDriverSwComponentType` .

⌋()

**[constr_4040] Synchronized mode groups must have same type** ⌈ `SwcBswSynchronizedModeGroupPrototype` can only refer to equally typed `ModeDeclarationGroupPrototype` s, i.e. which have identical `ModeDeclarationGroup` s.

⌋()

**[constr_4041] Synchronized mode groups must have same context** ⌈ The mapping defined by `SwcBswSynchronizedModeGroupPrototype` implies that the component providing the one mode group prototype is also mapped to the module which provides the other mode group prototype by means of synchronizing their respective behaviors in `SwcBswMapping` .

⌋()

**[constr_4042] Synchronized triggers must have same context** ⌈ The mapping defined by `SwcBswSynchronizedTrigger` implies that the component providing the one trigger is also mapped to the module which provides the other trigger by means of synchronizing their respective behaviors in `SwcBswMapping` .

⌋()

**[constr_4043] Period of `BswTimingEvent`** ⌈ `BswTimingEvent . period` shall be greater than 0.

⌋()

**[constr_4044] Content of `McSwEmulationMethodSupport`** ⌈ The following constraints hold for the attributes of `McSwEmulationMethodSupport` :

- If `category` is `DOUBLE_POINTERED` , a `baseReference` must exist.
- If `category` is `SINGLE_POINTERED` , a `referenceTable` must exist.

- If `category` is `INITIALIZED_RAM` , one or more `elementGroup` s must exist.

⌋()

**[constr_4045] `implementationConfigVariant` of preconfigured configuration** ⌈ An `EcucModuleConfigurationValues` element with the `implementationConfigVariant` set to the value `PreconfiguredConfiguration` shall only be referenced in the role `preconfiguredConfiguration` and no other value for `implementationConfigVariant` is allowed in this role.

⌋()

**[constr_4046] `implementationConfigVariant` of recommended configuration** ⌈ An `EcucModuleConfigurationValues` element with the `implementationConfigVariant` set to the value `RecommendedConfiguration` shall only be referenced in the role `recommendedConfiguration` and no other value for `implementationConfigVariant` is allowed in this role.

⌋()

**[constr_4047] Multiplicity of vendor specific configuration parameters** ⌈ The association `BswImplementation . vendorSpecificModuleDef` shall be implemented as reference to one or more instances of `EcucModuleDef` if the underlying `BswModuleDescription` has the `category` BSW_CLUSTER. In all other cases, it shall refer to exactly one instance of `EcucModuleDef` (the one belonging to this module).

⌋()

**[constr_4048] Multiplicity of preconfigured values** ⌈ The association `BswImplementation . preconfiguredConfiguration` shall be implemented as reference to zero or more different instances of `EcucModuleConfigurationValues` if the underlying `BswModuleDescription` has the `category` BSW_CLUSTER. In all other cases, it shall refer to at most one instance of `EcucModuleConfigurationValues` (the one belonging to this module).

⌋()

**[constr_4051] `RoleBasedDataAssignment` in BSW** ⌈ When used in the context of `BswServiceDependency` , the following restriction hold for date references described by `RoleBasedDataAssignment` :

- Within `RoleBasedDataAssignment . usedDataElement` , only the reference `AutosarVariableRef . localVariable` is applicable.
- Within `RoleBasedDataAssignment . usedParameterElement` , only the reference `AutosarParameterRef . localParameter` is applicable.
- The reference `RoleBasedDataAssignment . usedPim` shall not be set.

⌋()

**[constr_4052] `BswModuleEntry` returnType direction** ⌈
`BswModuleEntry . returnType . direction` must not have the value **in** or **inout** .

⌋()

**[constr_4053] `BswModuleEntry` argument direction** ⌈
If `BswModuleEntry . argument . direction` has the value **out** or **inout** , the corresponding `BswModuleEntry . argument . swDataDefProps` plus eventually referred `ImplementationDataType` must be such that they result in a pointer declaration.

⌋()

**[constr_4054] Unambiguous links to addressing method** ⌈ `MemorySection . executableEntity` must not be defined, if `MemorySection . swAddrMethod` represents a data section. `MemorySection . executableEntity` must not refer to an `ExecutableEntity` which is linked to a different `SwAddrMethod` than `MemorySection . swAddrMethod` .

⌋()

**[constr_4056] `BswModuleEntry` with no `returnType`** ⌈
In case of an empty return type (“void” in C) the reference `BswModuleEntry . returnType` shall not be set.

⌋()

**[constr_4057] `BswModuleEntry` with no argument** ⌈
In case of an empty argument list (“void” in C) no reference `BswModuleEntry . argument` shall be set.

⌋()

**[constr_4058] Different mode groups in mapped BSWM and SWC must have different names** ⌈ If an `SwcInternalBehavior` is mapped to a `BswInternalBehavior` the corresponding SWC and BSW module descriptions may not refer to different `ModeDeclarationGroup` s having the same `shortName` but different elements. This holds especially if these mode groups are not synchronized but used independently.

⌋()

**[constr_4059] Different mode groups referred by a BSWM must have different names** ⌈ A `BswModuleDescription` may not refer to different `ModeDeclarationGroup` s (via `requiredModeGroup` and/or `providedModeGroup` ) having the same `shortName` but different elements.

⌋()

**[constr_4060] Allowed values of `Trigger . swImplPolicy` for BSW** ⌈ The **only** allowed values for the attribute `Trigger . swImplPolicy` are either `STANDARD` (in which case the `Trigger` processing does not use a queue) or `QUEUED` (in which case the processing of `Trigger` s positively uses a queue).

⌋()

**[constr_4061] Completeness of MC emulation reference** ⌈ If an McDataInstance in the role of a subElement of another McDataInstance specifies an instanceInMemory , then the containing McDataInstance must also specify an instanceInMemory . The target of the latter (i.e. upper level) instanceInMemory must be identical (including array index, if defined) to the context of the first (i.e. lower level) instanceInMemory .

⌋()

**[constr_4062] Mandatory symbol for McDataInstance root** ⌈ McDataInstance s directly aggregated in McSupportData must have a valid McDataInstance . symbol .

⌋()

**[constr_4063] Restrictions of ModeRequestTypeMap in BSW** ⌈ For every ModeDeclarationGroup referenced by a ModeDeclarationGroupPrototype used in a BswModuleDescription a ModeRequestTypeMap shall exist that points to the ModeDeclarationGroup and also to an eligible ImplementationDataType .

The ModeRequestTypeMap shall be aggregated by a DataTypeMappingSet which is referenced from the BswInternalBehavior that is aggregated by the BswModuleDescription .

⌋()

**[constr_4064] Synchronized triggers must implement same policy** ⌈ The mapping defined by SwcBswSynchronizedTrigger is only valid if the attribute SwcBswSynchronizedTrigger . swcTrigger . swImplPolicy has the same value as the attribute SwcBswSynchronizedTrigger . bswTrigger . swImplPolicy .

⌋()

**[constr_4065] Allowed values of BswInternalTriggeringPoint . swImplPolicy** ⌈ The **only** allowed values for the attribute BswInternalTriggeringPoint . swImplPolicy are either STANDARD (in which case the internal trigger processing does not use a queue) or QUEUED (in which case the internal trigger processing uses a queue).

⌋()

**[constr_4066] BswModeSwitchEvent and the definition of ModeTransition** ⌈ For each pair of ModeDeclaration s referenced by a BswModeSwitchEvent with attribute activation set to onTransition a ModeTransition shall be defined in the corresponding direction (i.e. from exitedMode to enteredMode ). This constraint shall only apply if the respective ModeDeclarationGroup defines at least one modeTransition .

⌋()

**[constr_4067] Exclusive usage of data references in McFunctionDataRefSet** ⌈ The roles McFunctionDataRefSet . flatMapEntry and McFunctionDataRef-

Set . mcDataInstance shall be used exclusively within one McFunctionDataRefSet and one McFunction . This means, all instance of McFunctionDataRefSet aggregated by one McFunction shall use the same and only one of the two kinds of referencing their data.

⌋()

**[constr_4068] Semantics of McFunctionDataRefSet . flatInstanceDescriptor** ⌈

- An McFunctionDataRefSet aggregated in the role of McFunction . defCalprmSet or McFunction . refCalprmSet shall only refer to FlatInstanceDescriptor s that can be traced down to a ParameterDataPrototype and are declared for calibration access i.e. have an associated SwDataDefProps . swCalibrationAccess set to readWrite or readOnly .
- An McFunctionDataRefSet aggregated in the role of McFunction . inMeasurementSet , McFunction . outMeasurementSet or McFunction . locMeasurementSet shall only refer to FlatInstanceDescriptor s that can be traced down to either a VariableDataPrototype , an ArgumentDataPrototype or a ModeDeclarationGroupPrototype and are declared as measurable i.e. have an associated SwDataDefProps . swCalibrationAccess set to readOnly .

⌋()

**[constr_4069] Semantics of McFunctionDataRefSet . mcDataInstance** ⌈

- An McFunctionDataRefSet aggregated in the role of McFunction . defCalprmSet or McFunction . refCalprmSet shall only refer to McDataInstance s that are declared for calibration access i.e. are aggregated in the role McSupportData . mcParameterInstance .
- An McFunctionDataRefSet aggregated in the role of McFunction . inMeasurementSet , McFunction . outMeasurementSet or McFunction . locMeasurementSet shall only refer to McDataInstance s that are declared as measurable i.e. are aggregated in the role McSupportData . mcVariableInstance .

⌋()

**[constr_4070] Applicability of BswModuleEntity . activationReason** ⌈ An activationReason shall not be set

- for instances of BswInterruptEntity
- for instances of BswCalledEntity

⌋()

**[constr_4071] Synchronized runnables and schedulable entities must be consistent** ⌈ In the case that a RunnableEntity is mapped to a BswSchedulableEntity

the RTE Generator may emit an Entry Point Prototype for the RunnableEntity as well as an Entry Point Prototype for the BswSchedulableEntity (depending on the specified events for SWC resp. BSW). The SwcBswRunnableMapping instance controlling this case is only valid if several attributes of the mapped RunnableEntity and BswSchedulableEntity are consistent, especially all of the following constraints apply to the attributes of the given instance of SwcBswRunnableMapping :

- swcRunnable . symbol must be identical to the symbol of bswEntity as defined in TPS_BSWMDT_04138 .
- swcRunnable . minimumStartInterval must be identical to bswEntity . minimumStartInterval .
- swcRunnable . canBeInvokedConcurrently must be identical to bswEntity . implementedEntry . isReentrant .
- swcRunnable . swAddrMethod must either be empty or must have identical attributes as the SwAddrmethod defined via bswEntity . swAddrMethod . This is required to ensure a unique configuration for the memory segment of the underlying code entity.
- swcRunnable . activationReason and bswEntity . activationReason must have identical shortName if they define the same bitPosition and must have identical bitPosition if they define the same shortName

Please note also the SWS_RTE for further details.

⌋()

**[constr_4072] Constraints of SectionNamePrefix . implementedIn** ⌈

- The SectionNamePrefix and the DependencyOnArtifact connected via this link must belong to the same BswImplementation .
- The DependencyOnArtifact referred by this link must be aggregated by BswImplementation in the role requiredArtifact .
- The DependencyOnArtifact referred by this link must have the category value set to MEMMAP.

⌋()

**[constr_4073] McDataAccessDetails shall refer to one ECU Extract** ⌈ Within one given McDataAccessDetails , all instances of System referenced as the base of any McDataAccessDetails . role McDataAccessDetails or as the base of any McDataAccessDetails . role McDataAccessDetails shall be identical and of category ECU_EXTRACT .

⌋()

**[constr_4074] Compatibility of BswModuleClientServerEntry -s** ⌈ Two BswModuleClientServerEntry -s are compatible if and only if all of the following conditions hold:

- Their reentrancy values are identical. These values are taken from the attribute `isReentrant` or, if this is undefined, from `encapsulatedEntry . isReentrant` .
- Their synchronicity values are identical. These values are taken from the attribute `isSynchronous` or, if this is undefined, from `encapsulatedEntry . isSynchronous` .
- The two `BswModuleEntry` -s referred as `encapsulatedEntry` have completely identical attributes.

⌋()

**[constr_4075] Constraints for `providedData` and `requiredData`** ⌈ Sender-Receiver communication in BSW is restricted to the pattern of so-called *explicit communication* (in the same way as described for software components in [3] ) with queued behavior. This leads to some constraints for the `VariableDataPrototype` referred in the role `BswModuleDescription . providedData` or `BswModuleDescription . requiredData` :

- It shall not have an `initValue` .
- Its `swDataDefProps . swImplPolicy` shall be set to `queued` .
- Its `swDataDefProps . calibrationAccess` shall be set to `notAccessable` .

There are no further formal constraints on the attributes of the `VariableDataPrototype` to be used in these roles or on the underlying `AutosarDataPrototype` .

⌋()

**[constr_4076] Constraints on `BswModuleEntry` used for Client-Server** ⌈ A `BswModuleEntry` used in the role `BswModuleClientServerEntry . encapsulatedEntry` must have attribute values as follows:

- `callType` must be `regular` or `callback` .
- `executionContext` must be `task` .

⌋()

**[constr_4077] Constraints for `BswModuleEntity . reentrancyLevel`** ⌈

- If the attribute `isReentrant` of a `BswModuleEntry` referred by an `BswModuleEntity` in the role `implementedEntry` has the value `true` , then the attribute `reentrancyLevel` of the same `BswModuleEntity` (if it exists) can only have the values `singleCoreReentrant` or `multiCoreReentrant` .
- If the attribute `isReentrant` of a `BswModuleEntry` referred by an `BswModuleEntity` in the role `implementedEntry` has the values `false` , then there are no retrictions for the values of the attribute `reentrancyLevel` of the same `BswModuleEntity` (if it exists).

⌋()

**[constr_4078] Consistent usage of `BswOperationInvokedEvent`** ⌈ The `BswCalledEntity` referred by the attribute `BswOperationInvokedEvent . startsOnEvent` shall refer to the same `BswModuleEntry` (via its atttribute `implementedEntry` ) as the `BswOperationInvokedEvent` (via its attribute `entry . encapsulatedEntry .`

⌋()

**[constr_4079] `calledEntry` constraints for client-server calls** ⌈

- The `BswModuleClientServerEntry` aggregated as `calledEntry` in a `BswSynchronousServerCallPoint` must have the attribute `isSynchronous = true .`
- The `BswModuleClientServerEntry` aggregated as `calledEntry` in a `BswAsynchronousServerCallPoint` must have the attribute `isSynchronous = false .`

⌋()

**[constr_4080] Existence of reception policy** ⌈ If a `VariableDataPrototype` is referred from a `dataReceivePoint` of any `BswModuleEntity` in a given `BswInternalBehavior` , then exactly one corresponding `BswDataReceptionPolicy` must by aggregated by this `BswInternalBehavior .`

⌋()

**[constr_4081] Mode group used by BSW mode manager error event** ⌈ The `ModeDeclarationGroupPrototype` used by `BswModeManagerErrorEvent` must be referred as `BswModuleDescription . providedModeGroup` by the same module.

⌋()

**[constr_4083] `BswDistinguishedPartition` shall be used only in the context of a particular `BswInternalBehavior`** ⌈ All instances of `BswEvent` , `BswModuleCallPoint` and `BswVariableAccess` which refer to a `BswDistinguishedPartition` shall belong to the same `BswInternalBehavior` that also aggregates the referred `BswDistinguishedPartition .`

⌋()

**[constr_4084] Consistency of references of `InternalBehavior`** ⌈ The `SwcInternalBehavior` referenced by `SwcBswMapping . SwcBehavior` in the `SwcBswMapping` determined by `SwcImplementation . swcBswMapping` shall be identical to the `SwcInternalBehavior` referenced by `SwcImplementation . behavior .`

⌋()

**[constr_4085] Consistency of references of `InternalBehavior`** ⌈ The `BswInternalBehavior` referenced by `SwcBswMapping . bswBehavior` in the `SwcB-`

swMapping determined by BswImplementation . swcBswMapping shall be identical to the BswInternalBehavior referenced by BswImplementation . behavior .

⌋()

**[constr_4086] invocation of ExecutableEntity s by direct function call dependent from BswExecutionContext** ⌈ For example, if we take the fourth column in table table_3a_PossibleInvocationAsDirectFunctionCall , the invocation of an ExecutableEntity with an interruptCat1 BswExecutionContext can be implemented with a direct function call if the BswExecutionContext of the caller BswModuleEntry is set to task , interruptCat2 , or interruptCat1 .

This applies to the invocation of a triggered ExecutableEntity by the SchM_Trigger , SchM_ActMain or Rte_Trigger APIs, or to the invocation of an OnEntry ExecutableEntity , OnTransition ExecutableEntity , OnExit ExecutableEntity or mode switch acknowledge ExecutableEntity by the SchM_Switch or Rte_Switch APIs. For more information about the technical terms refer to [4]

⌋()

**[constr_4087] Usage of category "MACRO"** ⌈
It is only allowed to use the category "MACRO" for SwServiceArg if the owning BswModuleEntry has its swServiceImplPolicy attribute set to macro.

⌋()

**[constr_4088] Existence of RoleBasedDataTypeAssignment . role vs. RoleBasedDataAssignment . role** ⌈ The usage of a RoleBasedDataTypeAssignment with attribute role set to the value temporaryRamBlock is only allowed if **no** RoleBasedDataAssignment defined with attribute role set to value default Value exists in the owning BswServiceDependency .

⌋()

**[constr_4089] Association callbackHeader is only applicable for BSW modules** ⌈ The association callbackHeader is only supported for codeDescriptor s of BswImplementation and only permitted to reference ServiceNeeds owned by BswServiceDependency .

⌋()

**[constr_4090] The callbackHeader reference has to be consistent with behavior reference** ⌈ The reference callbackHeader is only allowed to reference ServiceNeeds in the context of the BswServiceDependency which in turn is referenced by the BswImplementation behavior of the BswImplementation owning the codeDescriptor .

⌋()

**[constr_4091] `AccessCount . value` needs to be unambiguous** ⌈ AUTOSAR model shall define at most one `AccessCount . value` per `countProfile` for a specific `AbstractAccessPoint` .

⌋()

**[constr_4092] Number of `ErrorTracerNeeds` in `BswInternalBehavior`** ⌈ A BswInternalBehavior shall provide at most one `ErrorTracerNeeds` element.

⌋()

**[constr_4093] Entries linked to `BswModuleEntry` s shall have compatible signature** ⌈ Matching `BswModuleEntry` s according to TPS_BSWMDT_04130 are compatible if the following condtions are fullfilled:

- both or neither of them define a `returnType`
- when the `returnType` s are defined, the `SwServiceArg` s in the role `returnType` shall be compatible
- both define the same number of compatible arguments in same order

⌋()

**[constr_4094] compatibility of `SwServiceArg` in role `returnType`** ⌈ `SwServiceArg` in role `returnType` are compatible if they are identically typed

⌋()

**[constr_4095] Compatibility of `SwServiceArg` in role `argument`** ⌈ `SwServiceArg` in role `returnType` are compatible if:

- they are identically typed

and

- if both do have the same `shortName`

⌋()

**[constr_4096] Matching `BswModuleEntry` s should have compatible attributes** ⌈ Matching `BswModuleEntry` s according to TPS_BSWMDT_04130 should be defined with identical values of the attributes

- `callType`
- `executionContext`
- `isReentrant`
- `isSynchronous`
- `serviceId`
- `swServiceImplPolicy`

- bswEntryKind

⌋()

**[constr_4097] Limitation on the number of BswExclusiveAreaPolicy s** ⌈ An ExclusiveArea can only be referenced by at most one BswExclusiveAreaPolicy .

⌋()

**[constr_4098] No mode disabling for BswOperationInvokedEvent** ⌈ A BswOperationInvokedEvent shall not have a reference to a ModeDeclaration in the role disabledInMode .

⌋()

## 2.11 TPS_DiagnosticExtractTemplate

**[constr_1324] Existence of attribute DiagnosticDataIdentifier . representsVin** ⌈ Within the context of a given DiagnosticContributionSet , the attribute DiagnosticDataIdentifier . representsVin shall have the value true for only a single DiagnosticDataIdentifier .

⌋()

**[constr_1325] Allowed attributes of SwDataDefProps for DiagnosticDataElement . swDataDefProps** ⌈ The allowed attributes of SwDataDefProps for the aggregation in the role DiagnosticDataElement . swDataDefProps are defined in table table_3a_SwDataDefPropsForDiagnosticDataElement .

⌋()

**[constr_1326] Existence of a variable-sized array** ⌈ The value of the attribute DiagnosticDataElement . arraySizeSemantics **shall not** be set to ArraySizeSemanticsEnum . variableSize if the respective DiagnosticDataElement is referenced from a DiagnosticServiceDataMapping .

⌋()

**[constr_1327] Multiplicity of DiagnosticEcuInstanceProps . ecuInstance** ⌈ The multiplicity of DiagnosticEcuInstanceProps . ecuInstance shall be limited to 1 and the enclosing DiagnosticContributionSet shall only refer to at most one DiagnosticEcuInstanceProps if the enclosing DiagnosticContributionSet is of category DIAGNOSTICS_ECU_EXTRACT .

⌋()

**[constr_1328] Consistency of DiagnosticEcuInstanceProps . ecuInstance and DiagnosticServiceTable . ecuInstance** ⌈ Each DiagnosticServiceTable referenced by any given DiagnosticContributionSet in the role serviceTable shall define a reference in the role DiagnosticServiceTable .

ecuInstance to an EcuInstance that is also referenced in the role DiagnosticEcuInstanceProps . ecuInstance by a DiagnosticEcuInstanceProps referenced by the mentioned DiagnosticContributionSet .

⌋()

**[constr_1329] Existence of concrete sub-classes of DiagnosticServiceClass in the context created by a DiagnosticContributionSet** ⌈ One of the following mutually exclusive conditions shall apply for the existence of any concrete sub-class of DiagnosticServiceClass in the context created by a DiagnosticContributionSet :

- The subclass of DiagnosticServiceClass shall only appear once in the context created by a DiagnosticContributionSet
- If the subclass of DiagnosticServiceClass appears multiple times in the context created by a DiagnosticContributionSet then all instances shall have identical values for all of their attributes.

  In case of aggregations the number of aggregated elements shall be identical and the values of primitive attributes of aggregated elements shall again be identical.

⌋()

**[constr_1330] Custom service identifier shall not overlap with standardized service identifiers** ⌈ The value of the attribute customServiceId shall not be set to any of the values reserved for standardized service identifiers as defined by the ISO 14229-1, see [1] .

⌋()

**[constr_1331] Existence of DiagnosticEcuReset . customSubFunctionNumber** ⌈ The attribute DiagnosticEcuReset . customSubFunctionNumber shall only exist if the value of DiagnosticEcuReset . category is outside the standardized set of values as defined by TPS_DEXT_01056 .

⌋()

**[constr_1332] Value range for DiagnosticEcuReset . customSubFunctionNumber** ⌈ The allowed value for DiagnosticEcuReset . customSubFunctionNumber shall always be within the closed interval **0x40** .. **0x7E** .

⌋()

**[constr_1333] Existence of DiagnosticMemoryIdentifier . memoryLowAddress and DiagnosticMemoryIdentifier . memoryHighAddress** ⌈ The attributes DiagnosticMemoryIdentifier . memoryLowAddress as well as DiagnosticMemoryIdentifier . memoryHighAddress shall not exist if the DiagnosticMemoryIdentifier referenced in the role memoryRange is referenced by a DiagnosticRequestDownload or a DiagnosticRequestUpload .

⌋()

**[constr_1334] Existence of `DiagnosticComControl . customSubFunction-Number`** ⌈ The attribute `DiagnosticComControl . customSubFunctionNumber` shall only exist if the value of `DiagnosticComControl . category` is outside the standardized set of values as defined by TPS_DEXT_01057 .

⌋()

**[constr_1335] Possible values for `DiagnosticComControl . customSubFunc-tionNumber`** ⌈ Given the fulfillment of constr_1334 , the value of a given `Diagnos-ticComControl . customSubFunctionNumber` shall always be within the closed interval **0x40** .. **0x5F** (for manufacturer-specific sub-functions) or the closed interval **0x60** .. **0x7E** (for supplier-specific sub-functions).

⌋()

**[constr_1336] Applicable value range for `DiagnosticComControlSpecific-Channel . subnetNumber`** ⌈ The value of attribute `DiagnosticComControl-SpecificChannel . subnetNumber` shall be within the closed interval **1** .. **14** .

⌋()

**[constr_1337] Allowed value range for attribute `DiagnosticComControlSubN-odeChannel . subNodeNumber`** ⌈ The value of attribute `DiagnosticComControl-SubNodeChannel . subNodeNumber` shall not exceed the closed interval **0** .. **65535** .

⌋()

**[constr_1338] Maximum number of aggregated `DiagnosticReadDataByPeri-odicIDClass . periodicRate`** ⌈ The number of aggregated `periodicRate` within the context of one `DiagnosticReadDataByPeriodicIDClass` shall be within the closed interval 1..3.

⌋()

**[constr_1339] Existence of `DiagnosticRoutine . start`** ⌈ In a complete `Diag-nosticExtract` , the attribute `DiagnosticRoutine . start` shall always exist for any given `DiagnosticRoutine` .

⌋()

**[constr_1340] Consistency of `DiagnosticServiceSwMapping` with respect to synchronously called `DiagnosticRoutine s`** ⌈ Each `DiagnosticSer-viceSwMapping` that references a `DiagnosticRoutineControl` that only aggre-gates a `DiagnosticStartRoutine` in the role `start` shall only reference a `Swc-ServiceDependency` or `BswServiceDependency` that in turn aggregates a `Diag-nosticRoutineNeeds` with attribute `diagRoutineType` set to `DiagnosticRou-tineTypeEnum . synchronous` .

⌋()

**[constr_1341] Consistency of `DiagnosticServiceSwMapping` with respect to asynchronously called `DiagnosticRoutine s`** ⌈ Each `DiagnosticServiceSwMapping` that references a `DiagnosticRoutineControl` that aggregates a `DiagnosticStopRoutine` and/or `DiagnosticRequestRoutineResults` in the role `stop` resp. `requestResult` shall only reference a `SwcServiceDependency` or `BswServiceDependency` that in turn aggregates a `DiagnosticRoutineNeeds` with attribute `diagRoutineType` set to `DiagnosticRoutineTypeEnum . asynchronous` .

⌋()

**[constr_1342] Possible values for `DiagnosticSecurityAccess . requestSeedId`** ⌈ The value of the attribute `DiagnosticSecurityAccess . requestSeedId` **shall only be set to an odd number** The even numbers are reserved for the identification of the corresponding *sendKey* sub-function, as explained by TPS_DEXT_01036 . .

The supported value range consists of the following list:

- all odd numbers in the closed interval **0x01** .. **0x41**
- **0x5F** (this corresponds to the case of *end-of-life activation of on-board pyrotechnic devices according to ISO 26021-2 [5]* )
- all odd numbers in the closed interval **0x61** .. **0x7E**

⌋()

**[constr_1343] Simultaneous existence of the attributes `DiagnosticServiceDataMapping . diagnosticDataElement` and `DiagnosticDataByIdentifier . dataIdentifier`** ⌈ A `DiagnosticServiceDataMapping . diagnosticDataElement` shall also be aggregated by a `DiagnosticDataByIdentifier` in the role `dataIdentifier . dataElement . dataElement` .

⌋()

**[constr_1344] Condition for the identification of data types of attributes `DiagnosticServiceDataMapping . mappedDataElement` and `DiagnosticServiceDataMapping . diagnosticDataElement`** ⌈ Both `DiagnosticServiceDataMapping . mappedDataElement` and `DiagnosticServiceDataMapping . diagnosticDataElement` shall be typed by either of the following options:

- `ApplicationPrimitiveDataType` where the value of attribute `category` is set to `VALUE` .
- `ImplementationDataType` where the value of attribute `category` is set to `VALUE` or to `TYPE_REFERENCE` that eventually resolves to an `ImplementationDataType` where attribute `category` is set to `VALUE` .

⌋()

**[constr_1345] `DiagnosticDataElement` shall not (finally) be aggregated by a `DiagnosticRoutine`** ⌈ A `DiagnosticDataElement` that is referenced by a `DiagnosticServiceDataMapping` shall not (finally) be aggregated by a `DiagnosticRoutine` .

⌋()

**[constr_1346] Allowed values of `DiagnosticServiceSwMapping . serviceInstance`** ⌈ The applicability of the `DiagnosticServiceSwMapping` is limited to predefined set of diagnostic services.

By regulation of the AUTOSAR standard, `DiagnosticServiceSwMapping . serviceInstance` shall only point to the following sub-classes of `DiagnosticServiceInstance` :

- `DiagnosticRoutine`
- `DiagnosticSecurityAccess`
- `DiagnosticReadDataByIdentifier`
- `DiagnosticWriteDataByIdentifier`
- `DiagnosticIOControl`

⌋()

**[constr_1347] Existence of attributes of `DiagnosticServiceSwMapping`** ⌈ For any given `DiagnosticServiceSwMapping` , **one and only one of** the following references shall exist:

- `DiagnosticServiceSwMapping . mappedFlatSwcServiceDependency`
- `DiagnosticServiceSwMapping . mappedSwcServiceDependencyInSystem`
- `DiagnosticServiceSwMapping . mappedBswServiceDependency`

⌋()

**[constr_1349] Value of `udsDtcValue` shall be unique** ⌈ The value of `udsDtcValue` shall be unique to any other DTC and DTC group value.

⌋()

**[constr_1350] Value of `DiagnosticTroubleCodeGroup . groupNumber` shall be unique** ⌈ The value of `DiagnosticTroubleCodeGroup . groupNumber` shall be unique to any other DTC and DTC group value.

⌋()

**[constr_1351] Value of `DiagnosticTroubleCodeGroup . groupNumber`** ⌈ To be compliant to ISO, the value of `DiagnosticTroubleCodeGroup . groupNumber` shall be set as defined in ISO 14229-1 [1] .

⌋()

**[constr_1352] Existence of `maxNumberFreezeFrameRecords` vs. `freezeFrame`** ⌈ If the attribute `DiagnosticTroubleCodeProps . maxNumberFreezeFrameRecords` exists than the attribute `DiagnosticTroubleCodeProps . freezeFrame` shall not exist or vice versa.

⌋()

**[constr_1353] Applicability of [1352]** ⌈ constr_1352 shall apply in the identical way (either one or the other attribute shall exist) for all `DiagnosticTroubleCodeProps` within the context of all `DiagnosticContributionSet` s of `category DIAGNOSTIC_ECU_EXTRACT` that refer to the same `EcuInstance` .

⌋()

**[constr_1354] Existence of attribute `DiagnosticTroubleCodeProps . freezeFrameContent`** ⌈ If one of the attributes `DiagnosticTroubleCodeProps . maxNumberFreezeFrameRecords` or `DiagnosticTroubleCodeProps . freezeFrame` exists then the attribute `DiagnosticTroubleCodeProps . freezeFrameContent` shall exist.

⌋()

**[constr_1355] Value of `extendedDataRecord . recordNumber`** ⌈ To be compliant to ISO, the value of `extendedDataRecord . recordNumber` shall be set in the interval as defined in ISO 14229-1 [1] .

⌋()

**[constr_1357] Value of `freezeFrame . recordNumber`** ⌈ To be compliant to ISO, the value of `freezeFrame . recordNumber` shall be set in the interval as defined in ISO 14229-1 [1] .

⌋()

**[constr_1359] Existence of attribute `DiagnosticDebounceAlgorithmProps . debounceCounterStorage`** ⌈ The attribute `DiagnosticDebounceAlgorithmProps . debounceCounterStorage` shall only exist if the aggregation `DiagnosticDebounceAlgorithmProps . debounceAlgorithm` actually aggregates a `DiagEventDebounceCounterBased`

⌋()

**[constr_1360] Usage of `DiagEventDebounceMonitorInternal` is not supported in the context of `DiagnosticDebounceAlgorithmProps`** ⌈ The usage of the meta-class `DiagEventDebounceMonitorInternal` for the aggregation in the role `DiagnosticDebounceAlgorithmProps . debounceAlgorithm` is not permitted.

⌋()

**[constr_1361] Number of `DiagnosticEventToEnableConditionGroupMapping` elements per `DiagnosticEvent`** ⌈ The mapping element `DiagnosticEventToEnableConditionGroupMapping` shall be created no more than once per `DiagnosticEvent` .

If several `DiagnosticEventToEnableConditionGroupMapping` elements referring the same `DiagnosticEvent` are defined, then the `Enable Condition Group` mapping shall be regarded as defective.

⌋()

**[constr_1362] Number of `DiagnosticEventToStorageConditionGroupMapping` elements per `DiagnosticEvent`** ⌈ The mapping element `DiagnosticEventToStorageConditionGroupMapping` shall be created no more than once or once per `DiagnosticEvent` .

If several `DiagnosticEventToStorageConditionGroupMapping` elements referring the same `DiagnosticEvent` are defined, then the `Storage Condition Group` mapping shall be regarded as defective.

⌋()

**[constr_1365] Multiplicity of `DiagnosticResponseOnEvent.event`** ⌈ The multiplicity of `DiagnosticResponseOnEvent.event` shall not exceed the upper bound **255** .

⌋()

**[constr_1366] Event ID in the context of diagnostic service `ResponseOnEvent` shall be unique** ⌈ The value of `DiagnosticResponseOnEvent.event.dataIdentifier.id` shall be unique within the context of a given `DiagnosticResponseOnEvent` .

⌋()

**[constr_1376] Multiplicity of reference `DiagnosticTroubleCodeProps.memoryDestination`** ⌈ For every given `DiagnosticTroubleCodeProps` , the reference in the role `DiagnosticTroubleCodeProps.memoryDestination` **shall not exceed** the upper multiplicity 2. constr_1377 applies.

⌋()

**[constr_1377] Existence of reference `DiagnosticTroubleCodeProps.memoryDestination`** ⌈ The reference `DiagnosticTroubleCodeProps.memoryDestination` shall **only** have the upper multiplicity 2 **if one (and only one) of** the referenced `DiagnosticTroubleCodeProps.memoryDestination` is a `DiagnosticMemoryDestinationMirror` .

⌋()

**[constr_1378] Value of `DiagnosticMemoryDestinationUserDefined.memoryId`** ⌈ Within the scope of one `DiagnosticContributionSet` , no two (or more)

DiagnosticMemoryDestinationUserDefined s shall exist that share the same value for attribute DiagnosticMemoryDestinationUserDefined . memoryId

⌋()

**[constr_1379] Existence of DiagnosticMemoryDestinationPrimary** ⌈ Within the scope of one DiagnosticContributionSet only one DiagnosticMemory-DestinationPrimary shall exist.

⌋()

**[constr_1380] Existence of DiagnosticMemoryDestinationMirror** ⌈ Within the scope of one DiagnosticContributionSet only one DiagnosticMemory-DestinationMirror shall exist.

⌋()

**[constr_1394] Value of DiagnosticDataElement . maxNumberOfElements depending on its existence** ⌈ If the attribute DiagnosticDataElement . maxNum-berOfElements exists then its value shall be greater than 0.

⌋()

**[constr_1405] Value of DiagnosticProtocol . serviceTable vs. Diagnos-ticServiceTable . protocolKind** ⌈ If the reference DiagnosticProtocol . serviceTable exists then the value of DiagnosticProtocol . protocolKind shall be identical to the value of DiagnosticServiceTable . protocolKind .

⌋()

**[constr_1406] DiagnosticServiceTable . diagnosticConnection vs. Diag-nosticProtocol . diagnosticConnection** ⌈ If a DiagnosticServiceTable exists that fulfills the following conditions:

- reference DiagnosticServiceTable . diagnosticConnection exists
- the DiagnosticServiceTable is referenced by means of DiagnosticPro-tocol . serviceTable

then all of the DiagnosticConnection s referenced by means of Diagnostic-ServiceTable . diagnosticConnection shall also be referenced in the role di-agnosticConnection from a DiagnosticProtocol that in turn references the respective DiagnosticServiceTable in the role DiagnosticProtocol . ser-viceTable .

⌋()

**[constr_1411] Existence of DiagnosticMemoryIdentifier . memoryHighAd-dressLabel vs. DiagnosticMemoryIdentifier . memoryHighAddress** ⌈ At most **one** of the attributes in the following list shall exist:

- DiagnosticMemoryIdentifier . memoryHighAddressLabel
- DiagnosticMemoryIdentifier . memoryHighAddress

⌋()

**[constr_1412] Existence of `DiagnosticMemoryIdentifier . memoryLowAddressLabel` vs. `DiagnosticMemoryIdentifier . memoryLowAddress`** ⌈ At most **one** of the attributes in the following list shall exist:

- `DiagnosticMemoryIdentifier . memoryLowAddressLabel`
- `DiagnosticMemoryIdentifier . memoryLowAddress`

⌋()

**[constr_1419] Value of `DiagnosticSecurityLevel . accessDataRecordSize`** ⌈ If the attribute `DiagnosticSecurityLevel . accessDataRecordSize` exists then its value shall be greater than zero.

⌋()

**[constr_1421] Consistency of `DiagnosticDynamicallyDefineDataIdentifierClass . subfunction`** ⌈ The values of `DiagnosticDynamicallyDefineDataIdentifierClass . subfunction` shall not repeat, i.e. every value of `DiagnosticDynamicallyDefineDataIdentifierSubfunctionEnum` shall at most appear once in the `subfunction` attribute.

⌋()

**[constr_1435] Debouncing in the presence of a `DiagnosticEventPortMapping`** ⌈ If a `DiagnosticEventPortMapping` exists and the enclosed `DiagnosticEventPortMapping . diagnosticEvent` is also referenced by a `DiagnosticEventToDebounceAlgorithmMapping` then the concrete subclass of the respective `DiagnosticEventToDebounceAlgorithmMapping . debounceAlgorithm . debounceAlgorithm` shall be identical to the `DiagnosticEventPortMapping . swcServiceDependencyInSystem` / `swcFlatServiceDependency . serviceNeeds . diagEventDebounceAlgorithm .`

It is assumed that the `DiagnosticEventPortMapping . swcServiceDependencyInSystem` / `swcFlatServiceDependency . serviceNeeds` is a `DiagnosticEventNeeds .`

⌋()

**[constr_1447] Restrictions for the value of `DiagnosticParameterIdentifier . id`** ⌈ The values 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, and 0xE0 are not allowed to appear in the value of `DiagnosticParameterIdentifier . id .`

⌋()

**[constr_1448] Interval of `DiagnosticParameterIdentifier . id`** ⌈ The allowed interval for values of `DiagnosticParameterIdentifier . id` shall not exceed [0..255].

⌋()

**[constr_1449] PID shall only carry a fixed-length collection of data** ⌈ The value of `DiagnosticParameterIdentifier . dataElement . dataElement . arraySizeSemantics` shall not be set to `variableSize` .

⌋()

**[constr_1450] Service mapping for ODB mode 0x01 for `DiagnosticParameterIdentifier`** ⌈ if a `DiagnosticServiceSwMapping` or `DiagnosticServiceDataMapping` refers to a `DiagnosticRequestCurrentPowertrainData` and a `DiagnosticDataElement` that is aggregated by a `DiagnosticParameterIdentifier` then then one of two alternative model configurations shall exist:

- `SwcServiceDependency` referenced by the same `DiagnosticServiceSwMapping` resp. `DiagnosticServiceDataMapping` shall aggregate an `ObdPidServiceNeeds` in the role `serviceNeeds` .
- The `BswServiceDependencyIdent` referenced by the same `DiagnosticServiceSwMapping` shall aggregate an `ObdPidServiceNeeds` in the role `serviceNeeds` .

⌋()

**[constr_1451] Service mapping for OBD mode 0x09 for `DiagnosticInfoType`** ⌈ if a `DiagnosticServiceSwMapping` refers to `DiagnosticRequestVehicleInfo` and a `DiagnosticDataElement` that is aggregated by a `DiagnosticInfoType` then one of two alternative model configurations shall exist:

- The `SwcServiceDependency` referenced by the same `DiagnosticServiceSwMapping` shall aggregate a `ObdInfoServiceNeeds` in the role `serviceNeeds` .
- The `BswServiceDependencyIdent` referenced by the same `DiagnosticServiceSwMapping` shall aggregate an `ObdInfoServiceNeeds` in the role `serviceNeeds` .

⌋()

**[constr_1452] Service mapping for OBD mode 0x08 for `DiagnosticInfoType`** ⌈ if a `DiagnosticServiceSwMapping` refers to a `DiagnosticRequestControlOfOnBoardDevice` then the `SwcServiceDependency` referenced by the same `DiagnosticServiceSwMapping` shall aggregate an `ObdControlServiceNeeds` in the role `serviceNeeds` .

⌋()

**[constr_1453] References from `DiagnosticFunctionInhibitSource`** ⌈ Each `DiagnosticFunctionInhibitSource` may either reference one of the following meta-classes in their respective roles:

- `DiagnosticFimAliasEventMapping` in the role `event`
- `DiagnosticFimAliasEventGroupMapping` in the role `eventGroup`

⌋()

**[constr_1454] `DiagnosticFimFunctionMapping` shall only reference a `SwcServiceDependency` that aggregates `FunctionInhibitionNeeds`** ⌈ A `DiagnosticFimFunctionMapping` shall only reference a `SwcServiceDependency` that aggregates `FunctionInhibitionNeeds` in the role `serviceNeeds` .

⌋()

**[constr_1455] Relation of `DiagnosticJ1939Node` to `J1939NmNode`** ⌈ Each `J1939NmNode` shall only be referenced in the role `nmNode` by a single `DiagnosticJ1939Node` .

⌋()

**[constr_1456] Valid interval for attribute `DiagnosticTroubleCodeJ1939 . fmi`** ⌈ The value of the attribute `DiagnosticTroubleCodeJ1939 . fmi` shall be in the interval 0..31.

⌋()

**[constr_1457] Service-only DTCs shall refer to a common memory section** ⌈ All `DiagnosticTroubleCodeJ1939` with attribute `kind` set to the value `serviceOnly` that reference the same `DiagnosticJ1939Node` shall also reference the same `DiagnosticTroubleCodeProps . memoryDestination` .

⌋()

**[constr_1458] Reference to `DiagnosticMemoryDestination`** ⌈ A `DiagnosticMemoryDestination` that is referenced by a `DiagnosticTroubleCodeJ1939 . dtcProps . memoryDestination` where the value of attribute `DiagnosticTroubleCodeJ1939 . kind` is set to `serviceOnly` shall **not be referenced by any other** `DiagnosticTroubleCodeJ1939` where attribute `kind` is set to any other value than `serviceOnly` .

⌋()

**[constr_1459] Existence of attributes of `DiagnosticTroubleCodeProps`** ⌈ The following list of attributes of meta-class `DiagnosticTroubleCodeProps` are not required and therefore shall be ignored if the `DiagnosticTroubleCodeProps` is referenced in the role `dtcProps` from a `DiagnosticTroubleCodeObd` :

- `freezeFrame`
- `freezeFrameContent`
- `memoryDestination`
- `extendedDataRecord`
- `aging`

⌋()

**[constr_1460] Restrictions for the value of `DiagnosticInfoType.id`** ⌈ The values 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, and 0xE0 are not allowed to appear in the value of `DiagnosticInfoType.id`.

⌋()

**[constr_1461] Restrictions for the value of `DiagnosticTestRoutineIdentifier.id`** ⌈ The values 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, and 0xE0 are not allowed to appear in the value of `DiagnosticTestRoutineIdentifier.id`.

⌋()

**[constr_1462] Restrictions for the value of `DiagnosticTestResult.testIdentifier.id`** ⌈ The values 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, and 0xE0 are not allowed to appear in the value of `DiagnosticTestResult.testIdentifier.id`.

⌋()

**[constr_1464] Allowed value range of `DiagnosticEnvConditionFormula.nrcValue`** ⌈ The value of attribute `DiagnosticEnvConditionFormula.nrcValue` shall be limited to the interval [1..255].

⌋()

**[constr_1465] Allowed values of `compareType` in the context of a `DiagnosticEnvDataCondition`** ⌈ Within the context of a `DiagnosticEnvDataCondition` **all values** of `DiagnosticCompareTypeEnum` are supported for the inherited attribute `compareType`.

⌋()

**[constr_1466] Allowed values of `compareType` in the context of a `DiagnosticEnvModeCondition`** ⌈ Within the context of a `DiagnosticEnvDataCondition` **only a subset** of the values of `DiagnosticCompareTypeEnum` is supported for the inherited attribute `compareType`, namely:

- `DiagnosticCompareTypeEnum.isEqual`
- `DiagnosticCompareTypeEnum.isNotEqual`

⌋()

**[constr_1467] References in `DiagnosticEnvModeCondition`** ⌈ In a `DiagnosticEnvModeCondition` the reference `modeElement` shall only point to a `DiagnosticEnvModeElement` that is aggregated inside the same `DiagnosticEnvironmentalCondition` as the `DiagnosticEnvModeCondition` itself.

⌋()

**[constr_1470] Value of `DiagnosticParameter.bitOffset`** ⌈ The value of `DiagnosticParameter.bitOffset` shall only be set to a multiple of 8.

⌋()

**[constr_1471] Existence of `DiagnosticDataIdentifier.didSize`** ⌈ The attribute `DiagnosticDataIdentifier.didSize` **shall not exist** if the value of `DiagnosticDataIdentifier.id` is **outside the range 0xF400-0xF4FF** .

⌋()

**[constr_1472] Existence of `DiagnosticDataIdentifier.supportInfoByte`** ⌈ The attribute `DiagnosticDataIdentifier.supportInfoByte` **shall not exist** if the value of `DiagnosticDataIdentifier.id` is **outside the range 0xF400-0x F4FF** .

⌋()

**[constr_1509] `extendedDataRecord.recordNumber` shall be unique within primary fault memory** ⌈ For all `DiagnosticTroubleCodeProps` that refer to `DiagnosticMemoryDestinationPrimary` in the role `memoryDestination` there shall be no two `extendedDataRecord.recordNumber` with the same value.

⌋()

**[constr_1510] `extendedDataRecord.recordNumber` shall be unique within mirror fault memory** ⌈ For all `DiagnosticTroubleCodeProps` that refer to `DiagnosticMemoryDestinationMirror` in the role `memoryDestination` there shall be no two `extendedDataRecord.recordNumber` with the same value.

⌋()

**[constr_1511] `extendedDataRecord.recordNumber` shall be unique within user-defined fault memory** ⌈ For all `DiagnosticTroubleCodeProps` that refer to `DiagnosticMemoryDestinationUserDefined` in the role `memoryDestination` there shall be no two extendedDataRecord. `recordNumber` with the same value for any `DiagnosticMemoryDestinationUserDefined` referenced as `DiagnosticTroubleCodeProps.memoryDestination` with a given value of `memoryId` .

⌋()

**[constr_1512] `freezeFrame.recordNumber` shall be unique within primary fault memory** ⌈ For all `DiagnosticTroubleCodeProps` that refer to `DiagnosticMemoryDestinationPrimary` in the role `memoryDestination` there shall be no two `freezeFrame.recordNumber` with the same value.

⌋()

**[constr_1513] `freezeFrame.recordNumber` shall be unique within mirror fault memory** ⌈ For all `DiagnosticTroubleCodeProps` that refer to `DiagnosticMemoryDestinationMirror` in the role `memoryDestination` there shall be no two `freezeFrame.recordNumber` with the same value.

⌋()

**[constr_1514] `freezeFrame.recordNumber` shall be unique within user-defined fault memory** ⌈ For all `DiagnosticTroubleCodeProps` that refer to

DiagnosticMemoryDestinationUserDefined in the role memoryDestination there shall be no two freezeFrame . recordNumber with the same value for any DiagnosticMemoryDestinationUserDefined referenced as DiagnosticTroubleCodeProps . memoryDestination with a given value of memoryId .

⌋()

**[constr_1515] Reference from DiagnosticRoutineControl to DiagnosticAccessPermission has no meaning** ⌈ The reference from DiagnosticRoutineControl (via its abstract base class DiagnosticServiceInstance ) in the role accessPermission to meta-class DiagnosticAccessPermission shall not be used.

⌋()

## 2.12 TPS_ECUConfiguration

**[constr_3022] EcucModuleDef category restriction** ⌈ The category definition shall be restricted to exactly the two defined ones:

- VENDOR_SPECIFIC_MODULE_DEFINITION
- STANDARDIZED_MODULE_DEFINITION

⌋()

**[constr_3023] Usage of apiServicePrefix** ⌈ The attribute apiServicePrefix is mandatory for VSMDs derived from the CDD StMD. The attribute shall not be provided for VSMDs derived from any other StMDs.

⌋()

**[constr_3091] Multiplicity of valueConfigClass** ⌈ The multiplicity of the attribute EcucCommonAttributes . valueConfigClass shall not exceed 3.

⌋()

**[constr_3092] Usage of configVariant and configClass attributes** ⌈ configVariant and configClass shall always exist as a pair for each existing EcucAbstractConfigurationClass ( EcucValueConfigurationClass or EcucMultiplicityConfigurationClass depending on the context).

⌋()

**[constr_3119] Necessary content of EcucDestinationUriDef s that are referenced by an EcucContainerDef** ⌈ The EcucDestinationUriDef that is referenced by the EcucContainerDef in the role destinationUri shall define at least the analogous set of container s, parameter s and reference s defined by the EcucDestinationUriPolicy of the EcucDestinationUriDef that is referenced by the EcucUriReferenceDef that targets the EcucContainerDef .

⌋()

**[constr_3120] Applicable attributes when `destinationUriNestingContract` is set to `targetContainer`** ⌈ If the `destinationUriNestingContract` is set to `targetContainer` the attributes `parameter` and `reference` shall not exist.

⌋()

**[constr_3200] Restriction on values of `EcucDefinitionElement . relatedTraceItem` in the VSMD** ⌈ The value of `EcucDefinitionElement . relatedTraceItem` in the VSMD shall never start with 'ECUC_'.

⌋()

**[constr_3217] Symbolic name reference shall point only to containers with a symbolic name value defined** ⌈ If an `EcucReferenceValue` exists that refers in the role `definition` to an `EcucAbstractInternalReferenceDef` with the attribute `requiresSymbolicNameValue` set to true, then the `EcucContainerValue` that is the target of the reference shall refer to an `EcucParamConfContainerDef` in the role `definition` that contains a definition of an `EcucParameterDef` where the attribute `symbolicNameValue` exists and is set to true. The `EcucContainerValue` shall define an `EcucParameterValue` that refers to an `EcucParameterDef` where the attribute `symbolicNameValue` exists and is set to true.

⌋()

**[constr_3228] `EcucSymbolicNameReferenceDef` presupposes `requiresSymbolicNameValue` set to true** ⌈ For `EcucSymbolicNameReferenceDef` the attribute `requiresSymbolicNameValue` shall always be set to true.

⌋()

**[constr_3233] `EcucModuleDef` that relies on `EcucCommonAttributes` with `valueConfigClass` set to `Link` / `PostBuild` of another `EcucModuleDef`** ⌈ If one `EcucModuleDef` relies on the `EcucCommonAttributes` (parameters and references) with `valueConfigClass . configClass` set to `Link` / `PostBuild` of another `EcucModuleDef` , the values of these `EcucCommonAttributes` can only be changed at `Link` / `PostBuild` time if the corresponding `EcucModuleConfigurationValues` of the using `EcucModuleDef` has the `implementationConfigVariant` set to `VariantLinkTime` / `VariantPostBuild` , respectively.

⌋()

**[constr_3234] `EcucModuleDef` that relies on `EcucCommonAttributes` with `multiplicityConfigClass` set to `Link` / `PostBuild` of another `EcucModuleDef`** ⌈ If one `EcucModuleDef` relies on the `EcucCommonAttributes` (parameters and references) with `multiplicityConfigClass . configClass` set to `Link` / `PostBuild` of another `EcucModuleDef` , the number of instances of these `EcucCommonAttributes` can only be changed at `Link` / `PostBuild` time if the corresponding `EcucModuleConfigurationValues` of the using `EcucModuleDef` has

the implementationConfigVariant set to VariantLinkTime / VariantPostBuild , respectively.

⌋()

**[constr_3235] EcucModuleDef that relies on EcucContainerDef s with multiplicityConfigClass set to Link / PostBuild of another EcucModuleDef** ⌈ If one EcucModuleDef relies on the EcucContainerDef s with multiplicityConfigClass . configClass set to Link / PostBuild of another EcucModuleDef , the number of instances of these EcucContainerDef s can only be changed at Link / PostBuild time if the corresponding EcucModuleConfigurationValues of the using EcucModuleDef has the implementationConfigVariant set to VariantLinkTime / VariantPostBuild , respectively.

⌋()

**[constr_3236] EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantValue set to true of another EcucModuleDef** ⌈ If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildVariantValue set to true of another EcucModuleDef , the values of these EcucCommonAttributes can only differ in different post-build variants if the implementation of the using EcucModuleDef supports post-build variations.

⌋()

**[constr_3237] EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantMultiplicity set to true of another EcucModuleDef** ⌈ If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildVariantMultiplicity set to true of another EcucModuleDef , the number of instances of these EcucCommonAttributes can only differ in different post-build variants if the implementation of the using EcucModuleDef supports post-build variations.

⌋()

**[constr_3238] EcucModuleDef that relies on EcucContainerDef with postBuildVariantMultiplicity set to true of another EcucModuleDef** ⌈ If one EcucModuleDef relies on the EcucContainerDef s with postBuildVariantMultiplicity set to true of another EcucModuleDef , the number of instances of these EcucContainerDef s can can only differ in different post-build variants if the implementation of the using EcucModuleDef supports post-build variations.

⌋()

**[constr_3307] ShortNames of PredefinedVariant s referenced by EcucPostBuildVariantRef s** ⌈ All PredefinedVariant s that are referenced by EcucPostBuildVariantRef s shall have different shortName s.

⌋()

**[constr_3509] Applicability of `scope` attribute** ⌈ The usage of the attribute `scope` is prohibited for `EcucModuleDef` and for sub-classes of `EcucContainerDef` (i.e. `EcucChoiceContainerDef` and `EcucParamConfContainerDef` ).

⌋()

**[constr_5015] Multiplicity of `multiplicityConfigClass`** ⌈ The multiplicity of the attribute `EcucCommonAttributes . multiplicityConfigClass` shall not exceed 3.

⌋()

**[constr_5500] Applicability of the `multiplicityConfigClass` attribute** ⌈ The `multiplicityConfigClass` attribute is applicable only to `EcucContainerDef` s which have `upperMultiplicity` greater than `lowerMultiplicity` .

⌋()

**[constr_5502] Introduction of new `EcucParameterValue` s of type `EcucFunctionNameDef` at post-build time** ⌈ In case a new `EcucParameterValue` s of type `EcucFunctionNameDef` (see Chapter `sec_3a_ParamDefFunctionName` ) is introduced at post-build time, it's value shall be one of the existing function names (e.g. callouts). This means that it is not allowed to introduce new functions at post-build time.

⌋()

**[constr_5504] Removing an instance of the `EcucContainerDef` at post-build time** ⌈ Only instances of `EcucContainerDef` s with `multiplicityConfigClass . configClass` set to `PostBuild` in the `multiplicityConfigClass . configVariant VariantPostBuild` which are not referenced or are exclusively referenced by `EcucAbstractReferenceDef` s with `valueConfigClass . configClass` set to `PostBuild` in the `valueConfigClass . configVariant VariantPostBuild` and have been introduced at post-build time (not part of the initial configuration before post-build updates) can be removed at post-build time.

⌋()

**[constr_5505] Configuration class of the elements of the `EcucQueryExpression`** ⌈ The elements of the `EcucQueryExpression` involved in one calculation formula shall have lower or equal configuration class (where `PreCompile` configuration class is considered to be the lowest and `PostBuild` the highest) with respect to the context element in which the calculation is performed (e.g. a `Link` configuration parameter can not calculate its value based on a `PostBuild` parameters value).

⌋()

**[constr_5506] Applicability of `postBuildVariantMultiplicity` attribute** ⌈ The `postBuildVariantMultiplicity` attribute of `EcucContainerDef` is applicable only to `EcucContainerDef` s which have `upperMultiplicity` greater than `lowerMultiplicity` .

⌋()

**[constr_5507] Value of `EcucContainerDef.postBuildVariantMultiplicity` if `postBuildVariantSupport` is set to `false`** ⌈ If `postBuildVariantSupport` is set to `false`, every `EcucContainerDef` in this `EcucModuleDef` with `upperMultiplicity` greater than `lowerMultiplicity` shall have its `postBuildVariantMultiplicity` attribute set to `false`.

⌋()

**[constr_5508] Applicability of `postBuildVariantMultiplicity` attribute** ⌈ The `postBuildVariantMultiplicity` attribute is applicable only to `EcucCommonAttributes` which have `upperMultiplicity` greater than `lowerMultiplicity`.

⌋()

**[constr_5509] Value of `postBuildVariantMultiplicity` if `postBuildVariantSupport` is set to `false`** ⌈ If `postBuildVariantSupport` is set to `false`, every `EcucCommonAttributes` in this `EcucModuleDef` with `upperMultiplicity` greater than `lowerMultiplicity` shall have its `postBuildVariantMultiplicity` attribute set to `false`.

⌋()

**[constr_5510] Value of `postBuildVariantValue` if `postBuildVariantSupport` is set to `false`** ⌈ If `postBuildVariantSupport` is set to `false`, every `EcucCommonAttributes` in this `EcucModuleDef` shall have its `postBuildVariantValue` attribute set to `false`.

⌋()

**[constr_5512] `postBuildVariantValue` attribute of `symbolicNameValue` parameters** ⌈ The values of `EcucParameterDef`s with `symbolicNameValue` attribute set to `true` shall have their `postBuildVariantValue` set to `false`.

⌋()

**[constr_5514] Applicability of the `multiplicityConfigClass` attribute** ⌈ The `multiplicityConfigClass` attribute is applicable only to `EcucCommonAttributes` which have `upperMultiplicity` greater than `lowerMultiplicity`.

⌋()

**[constr_5520] `valueConfigClass` attribute of `symbolicNameValue` parameters** ⌈ The values of `EcucParameterDef`s with `symbolicNameValue` attribute set to `true` shall have their `valueConfigClass.configClass` set to `PreCompile` for all `valueConfigClass.configVariant`s.

⌋()

**[constr_5521] `multiplicityConfigClass` attribute of `symbolicNameValue` parameters** ⌈ The values of `EcucParameterDef`s with `symbolicNameValue` at-

tribute set to `true` shall have their `multiplicityConfigClass . configClass` set to `PreCompile` for all `multiplicityConfigClass . configVariant` s.

⌋()

**[constr_5522] `postBuildVariantMultiplicity` attribute of `symbolicNameValue` parameters** ⌈ The values of `EcucParameterDef` s with `symbolicNameValue` attribute set to `true` shall have their `postBuildVariantMultiplicity` set to `false` .

⌋()

**[constr_5523] Allowed `configClass` es for paired `configVariant` s** ⌈ `PublishedInformation configClass` is supported by all `configVariant` s where TPS_ECUC_02071 applies. Additionally, `VariantPreCompile configVariant` supports `PreCompile configClass` , `VariantLinkTime configVariant` supports `PreCompile` and `Link configClass` es, and `VariantPostBuild configVariant` supports `PreCompile` , `Link` and `PostBuild configClass` es.

⌋()

## 2.13 TPS_ECUResourceTemplate

**[constr_3500] `category` of `HwAttributeDef` shall not be extended** ⌈ In contrast to the general rule that `category` can be extended by user-specific values it is **not allowed** to extend the meaning of the attribute `category` of meta-class `HwAttributeDef`

⌋()

**[constr_3511] `HwType` shall not have a reference to another `HwType`** ⌈ A `HwType` (being a `HwDescriptionEntity` ) shall not have a reference to another `HwType` in the role `hwType` . The definition of `HwType` s is not hierarchical.

⌋()

**[constr_3512] No support of multiple instantiation** ⌈ An essential constraint is that each `HwElement` can only be target of one `nestedElement` reference. This means that there is no concept of multiple instantiation of hardware elements. If the same hardware element shall be used several times (using the `nestedElement` reference) each occurrence has to have its own description. This is also true for nested elements of the referenced nested element.

⌋()

**[constr_3513] Scope of connections** ⌈ Each hardware connection shall only connect features which both are in the hierarchical scope of the hardware element. The hierarchical scope encloses

- all features belonging to the hardware element containing the connection

- all features belonging to hardware elements which are referenced directly and indirectly in the `nestedElement` relation from the hardware element containing connection.

⌋()

## 2.14 TPS_FeatureModelExchangeFormat

**[constr_5001] `FMFeatureRelation` shall not establish self-references** ⌈ A `FMFeatureRelation` that is aggregated by a `FMFeature` $f$ shall not reference $f$ in the role `feature` . In other words: self-references are not allowed.

⌋()

**[constr_5002] `FMFeatureSelectionSet` shall not have cycles in the `include` relation** ⌈ Let $S$ be a `FMFeatureSelectionSet` and let $G$ be the *inclusion graph* for all `FMFeatureSelectionSet` s as defined in TPS_FMDT_00032 . There shall be no cycles in the inclusion graph.

⌋()

**[constr_5003] `FMFeatureSelectionSet` shall not overwrite the state of included features** ⌈ Let $S$ be a `FMFeatureSelectionSet` that aggregates a `FMFeatureSelection` that has the `state` $s$ and which refers to a `FMFeature` $f$ in the role `feature` . Furthermore, let $S_1$ be a `FMFeatureSelectionSet` that aggregates a `FMFeatureSelection` that has the `state` $s_1$ and refers *to the same* `FMFeature` $f$ in the role `feature` . Finally assume that $S$ refers to $S_1$ in the role `include` .

Then the following conditions shall hold:

1. If the value of the attribute `state` of $s_1$ is `undecided` , then the value of the attribute `state` of $s$ may be one of `selected` , `deselected` , and `undecided` .
2. If the value of the attribute `state` of $s_1$ is `selected` or `deselected` , then the value of the attribute `state` of $s$ shall be the same as the attribute `state` in $s_1$ , or `undecided` .
3. Any other constellation is considered an error.

⌋()

**[constr_5005] `FMFeature` shall not be referenced from more than one `FMFeatureDecomposition`** ⌈ Let $f$ be a `FMFeature` that is referenced from a `FMFeatureDecomposition` in the role `feature` . Then no other `FMFeatureDecomposition` shall reference $f$ in the role `feature` .

⌋()

**[constr_5007] `FMFeature` shall only be referenced from one `FMFeatureModel` in the role `feature`** ⌈ Let $f$ be a `FMFeature` , and $F$ , $F'$ be `FMFeatureModel`

s where $F$ references $f$ in the role `feature` , and $F'$ also references $f$ in the role `feature` . Then $F = F'$ .

⌋()

**[constr_5008] If present, the root feature shall be part of the feature model** ⌈ Let $r$ be the `FMFeature` referenced from `FMFeatureModel` in the role `root` , and $\{f_1, f_2, \ldots, f_n\}$ the set of features referenced from the same `FMFeatureModel` in the role `feature` .

Then the following condition shall hold: $r \in \{f_1, f_2, \ldots, f_n\}$ .

⌋()

**[constr_5009] Root feature shall be present if and only if the feature model is not empty** ⌈ If a `FMFeatureModel` refers to one or more `FMFeature` elements in the role `feature` , then exactly one of them shall be referenced by `FMFeatureModel` in the role `root` .

On the contrary, if `FMFeatureModel` does not refer to any `FMFeature` s in the role `feature` , then `root` shall be empty.

⌋()

**[constr_5010] `FMFeatureDecomposition` may refer to a root feature of another feature model, but only once.** ⌈ Let $f_A$ be a `FMFeature` that is referenced by `FMFeatureModel` $A$ in the role `feature` , but is also referenced from a `FMFeature-Decomposition` that is aggregated by a `FMFeature` $f_B$ in the role `decomposition` .

Furthermore, let $B$ be the `FMFeatureModel` that references $f_B$ in the role `feature` with $A \neq B$ . That is, $f_A$ and $f_B$ belong to different feature models.

Then *both* the following conditions shall hold:

1. $f_A$ is referenced from $A$ in the role `root` .
2. There is no other `FMFeatureDecomposition` (neither in $B$ nor in any other `FMFeatureModel` ) that references $f_B$ in the role `feature` .

⌋()

**[constr_5011] `FMFormulaByFeaturesAndAttributes` can refer to `FMFeature` s and `FMAttributeDef` s, but not to system constants** ⌈ A formula of class `FM-FormulaByFeaturesAndAttributes` is an expression that can use `FMFeature` s and `FMAttributeDef` s, but is not allowed to use `SwSystemconst` s.

⌋()

**[constr_5013] Attributes `min` and `max` of `FMFeatureDecomposition` reserved for `category` `MULTIPLEFEATURE`** ⌈ The optional attributes `min` and `max` of `FMFeatureDecomposition` are only allowed to be present if the `category` of the `FMFeatureDecomposition` is `MULTIPLEFEATURE` .

⌋()

**[constr_5018] `FMFeatureSelectionSet` shall not include the same feature twice** ⌈ Let $\{s_1, s_2, \ldots, s_n\}$ be the set of `FMFeatureSelection` elements that are aggregated by a `FMFeatureSelectionSet` in the role `selection` . Furthermore, for each $s_i$ , let $f_i$ be the `FMFeature` that is referred to in the role `feature` . Then the following condition shall hold true:

$$\forall i, j \in \{1, 2, \ldots, n\} : i \neq j \Rightarrow f_i \neq f_j$$

⌋()

**[constr_5019] `FMFeatureModel` shall not contain the same `FMFeature` twice** ⌈ Let $F$ be a `FMFeatureModel` , and let $f, f'$ be `FMFeature` s that are referenced from $F$ in the role `feature` . Then $f \neq f'$ .

⌋()

**[constr_5020] Every `FMFeature` shall be contained in a `FMFeatureModel`** ⌈ For every `FMFeature` $f$ , there shall be a `FMFeatureModel` that refers to $f$ in the role `feature` .

⌋()

**[constr_5021] The underlying graph of a feature model shall be a tree.** ⌈ Let $F$ be a `FMFeatureModel` and $G$ be the underlying graph of $F$ as defined in TPS_FMDT_00034 . Then $G$ shall be a tree. Hence, we also refer to $G$ as the *underlying tree* of $F$ .

⌋()

**[constr_5022] The root feature of a `FMFeatureModel` refers to the root of the underlying tree.** ⌈ Let $F$ be a `FMFeatureModel` and $G$ be the underlying tree of $F$ as defined in TPS_FMDT_00034 . Furthermore, let $r$ be the `FMFeature` referred to by the `root` feature of the `FMFeatureModel` .

Then the node in $G$ which corresponds to $r$ is the root of the tree $G$ .

⌋()

**[constr_5023] `FMFeatureSelectionSet` may only refer to `FMFeature` s from the associated `FMFeatureModel`** ⌈ Let $S$ be a `FMFeatureSelectionSet` , and $\{f_1, f_2, \ldots, f_n\}$ be its *feature set* ( TPS_FMDT_00009 ). Furthermore, let $\{g_1, g_2, \ldots, g_m\}$ be the combined *feature sets* of the `FMFeatureModel` s to which $S$ refers to in the role `featureModel` .

Then the following condition shall hold: $\{f_1, f_2, \ldots, f_n\} \subseteq \{g_1, g_2, \ldots, g_m\}$ .

⌋()

**[constr_5024] `FMFeatureSelectionSet` shall not include itself** ⌈ Let $S$ be a `FMFeatureSelectionSet` and let $S'$ be the `FMFeatureSelectionSet` to which $S$ refers to in the role `include` .

Then the following condition shall hold: $S \neq S'$ .

⌋()

**[constr_5025] `FMFeatureSelectionSet` shall not overwrite the state of included features** ⌈ Let $S$ be a `FMFeatureSelectionSet` that aggregates a `FMFeatureSelection` that has the `state` $s$ and which refers to a `FMFeature` $f$ in the role `feature` . Furthermore, let $S_1$ ( $S_2$ ) be a `FMFeatureSelectionSet` that aggregates a `FMFeatureSelection` that has the `state` $s_1$ ( $s_2$ ) and refers *to the same* `FMFeature` $f$ in the role `feature` . Finally assume that $S$ refers to $S_1$ *and* $S_2$ in the role `include` .

Then the following conditions shall hold:

1. If the values of the attributes `state` of $s_1$ and $s_2$ are both `undecided` , then the value of the attribute `state` of $s$ may be `selected` , `deselected` or `undecided` .
2. If the value of the attribute `state` of $s_1$ is `undecided` and the value of the attribute `state` of $s_2$ is `selected` or `deselected` , then the value of the attribute `state` of $s$ shall be the same as the attribute `state` in $s_2$ , or `undecided` .
3. If the value of the attribute `state` of $s_2$ is `undecided` and the value of the attribute `state` of $s_1$ is `selected` or `deselected` , then the value of the attribute `state` of $s$ shall be the same as the attribute `state` in $s_1$ , or `undecided` .
4. If the values of the attributes `state` of $s_1$ and $s_2$ are both either `selected` or `deselected` , then the value of the attribute `state` of $s$ shall be the same as in attribute $s_1$ , or `undecided` .
5. Any other constellation is considered an error.

⌋()

**[constr_5026] Semantics of attributes `max` and `min` in class `FMAttributeDef`** ⌈ The following conditions shall hold for all instances of the class `FMAttributeDef` :

- `min` $\leq$ `defaultValue` $\leq$ `max` ( `min` and `max` are both closed intervals)
- `min` $<$ `defaultValue` $\leq$ `max` ( `min` is an open interval, `max` is a closed interval)
- `min` $<$ `defaultValue` $<$ `max` ( `min` and `max` are both open intervals)
- `min` $\leq$ `defaultValue` $<$ `max` ( `min` is a closed interval, `max` is an open interval)

⌋()

**[constr_5027] Semantics of attributes `max` and `min` of `FMAttributeDef` in class `FMAttributeValue`** ⌈ Let $v$ be the attribute `value` of an `FMAttributeValue` $V$ that refers to `FMAttributeDef` $D$ in the role `definition` . Furthermore, let $min$ and $max$ be the values of the attributes `min` and `max` of $D$ .

The following condition shall hold true:

$$min \leq v \leq max$$

⌋()

**[constr_5028] Only one `FMAttributeValue` per `FMAttributeDef`** ⌈ Let $S$ be a `FMFeatureSelectionSet` whose `FMFeatureSelection` s aggregate `FMAttributeValue` s $\{v_1, v_2, \ldots, v_n\}$ in the role `attributeValue` . For each $v_i$ , let $f_i$ be the `FMFeature` to which $v_i$ refers to in the role `attributeDef` . Then the following condition shall hold:

$$\forall i \in \{1, \ldots, n\} : i \neq j \Rightarrow f_i \neq f_j$$

⌋()

## 2.15 TPS_GenericStructureTemplate

**[constr_2501] Blueprint of blueprints are not supported** ⌈ Note that objects modeled particularly as a “blueprint” (e.g. `PortPrototypeBlueprint` ) also live in a package of category `BLUEPRINT` . Strictly speaking this means that they can be “blueprints” of “blueprints”. This indirection is not intended and not supported.

⌋()

**[constr_2502] Merged model shall be compliant to the meta-model** ⌈ A model merged from `atpSplitable` elements shall adhere to the consistency rules of the *pure meta model* . Note that the required lower multiplicities depend on the process phase therefore the AUTOSAR schema sets them mainly to 0. This also applies to the bound model.

⌋()

**[constr_2503] Bound model must be compliant to the pure meta model** ⌈ The *completely bound M1 model* Completely bound includes post build! must adhere to the *pure meta model* with respect to consistency rules and semantic constraints defined in the related template specifications. Especially, the multiplicities in the bound model must conform to the multiplicities and the constraints of the *pure meta model* .

⌋()

**[constr_2504] Constraint to `bindingTime`** ⌈ The tag `vh.latestBindingTime` *constraints* the value of the attribute `bindingTime` from TPS_GST_00190 . Hence, it defines the latest point in methodology which is allowed as value for `bindingTime` of this particular application of `atpVariation` .

⌋()

**[constr_2505] Multiplicity after binding** ⌈ *if* Phase $\geq$ `{partRole}.BindingTime` *then* number of `{partRole}` ’s = $n$

⌋()

**[constr_2506] Attributes in property set pattern** ⌈ On M1 level, let $C$ be the set of attributes (or aggregated elements The constraints defined in this section apply to attributes as well as aggregates elements, due to the close relationship of the two in the AUTOSAR meta model. For simplicity, the rest of this section talks about “attributes” only. ) that would have been in the original In this context, “original” means `{PropertySetClass}` without the stereotype `atpVariation` . In other words, “original” means “as in the pure meta model”. `{PropertySetClass}` object, and $C_1, \ldots, C_n$ be the respective sets of attributes in the `{PropertySetClass}Conditional` objects **for a given variant** . Also, let $C'$ be the set of non-optional attributes, e.g., those with a lower multiplicity of 1.

We define the following constraints:

$$\forall C_i, C_j \text{ in the given variant} : C_i \cap C_j = \emptyset$$
$$C' \subseteq C_1 \cup C_2 \cup \ldots C_n \subseteq C$$

⌋()

**[constr_2507] `EvaluatedVariantSet` shall not refer to itself** ⌈ An `EvaluatedVariantSet` shall not refer to itself directly or via other `EvaluatedVariantSet` .

⌋()

**[constr_2508] Name space of `shortName`** ⌈ The content of `shortName` needs to be unique (case insensitive) within a given `Identifiable` . Note that the check for uniqueness of `shortName` must be performed case insensitively. This supports the good practice that names should not differ in upper / lower case only which would cause a lot of confusion.

The term “case insensitive” indicates that the characters in the sets

```
{a b c d e f g h i j k l m n o p q r s t u v w x y z}
{A B C D E F G H I J K L M N O P Q R S T U V W X Y Z}
```

are respectively considered to be the same. In other words case-insensitive check for uniqueness of `shortName` s results in the fact that e.g. elements with `shortName` "X" and "x" are considered the same and shall **not** exist in the same package.

⌋()

**[constr_2509] `ReferenceBase` needs to be unique in a package** ⌈ The `shortLabel` of a reference base needs to be unique in (not within) a package. Note that it is not necessary to be unique within (to say in deeper levels) of a package.

⌋()

**[constr_2510] only one default `ReferenceBase`** ⌈ Only one `ReferenceBase` per level can be marked as default ( `default` =”true”).

⌋()

**[constr_2511] Named reference bases shall be available** ⌈ If there is a relative references, then one of the containing packages shall have a `referenceBase` with a `shortLabel` equal to the `base` of the reference.

⌋()

**[constr_2512] `shortName` uniqueness constraint for variants** ⌈ `shortName` + `shortLabel` of a variant element must be unique within the name space established by the surrounding `Identifiable` .

⌋()

**[constr_2514] `shortLabel` in `VariationPoint` must be unique** ⌈ The combination of `shortName` and `shortLabel` shall be unique within the next enclosing `Identifiable {WholeClass}` . In case the `shortName` does not exist on the `{PartClass}` the `shortLabel` is unnecessary. In case the `shortName` of the `{PartClass}` is unique in the context of the `{WholeClass}` the `shortLabel` is unnecessary.

⌋()

**[constr_2515] Categories of packages shall not conflict** ⌈ If a non empty category is defined for a package, then all sub packages shall have empty category or the same category. See table `table_3a_RulesCategoriesOfSubPackages` . Additionally, the "Rules for references between elements in packages with specific categories" shall apply. See table `table_3a_RulesReferencesBetweenElementsPackagesSpecCategories` .

⌋()

**[constr_2516] Return type of an `AttributeValueVariationPoint`** ⌈ When such a formula is evaluated by a software tool, and the return value of the formula is shall be compatible to the type of the attribute in the pure meta-model.

⌋()

**[constr_2517] `postbuildVariantCondition` only for PostBuild** ⌈ Aggregation of `PostBuildVariantCondition` in `VariationPoint` is only allowed if the annotated model states `vh.latestBindingTime` to `PostBuild` .

⌋()

**[constr_2518] Binding time is constrained** ⌈ Note that this binding time is again constrained by the value of the tag `vh.latestBindingTime` .

⌋()

**[constr_2519] `PredefinedVariant` s need to be consistent** ⌈ If a `PredefinedVariant` plus its `includedVariant` s references more than one `SwSystemconstantValueSet` all `value` attributes in `SwSystemconstValue` s for a particular `SwSystemconst` must be identical.

⌋()

**[constr_2520] Nesting of lists shall be limited** ⌈ The nesting of lists shall be limited to a reasonable depth such that it can safely be rendered on A4 pages. A reasonable approach is not to nest more than three levels.

⌋()

**[constr_2521] The `shortLabel` in `AttributeValueVariationPoint` shall be unique** ⌈ The `shortLabel` must be unique within the next enclosing `Identifiable` , and is used to individually address variation points in the *variant rich M1 model* .

⌋()

**[constr_2522] Notes should not be nested** ⌈ Note even if it is possible to nest notes it is not recommended to do so, since it might lead to problems with the rendering of the note icon.

⌋()

**[constr_2523] Used languages need to be consistent** ⌈ The used languages of an AUTOSAR file are specified in the top level `adminData` . All other elements shall be provided in the languages specified for the document.

⌋()

**[constr_2524] Non splitable elements in one file** ⌈ If the *aggregation* / *attribute* is **not** `atpSplitable` , then all aggregated element(s) shall be described in the same physical file as the aggregating element.

⌋()

**[constr_2525] Non splitable elements shall not be repeated** ⌈ Properties (namely aggregations, references and primitive attributes) which are not marked as `atpSplitable` must be all together in one physical file. They shall not be repeated in the split files unless they are an attribute which is used as a part of the splitkey. Another special case is handling of `atpStructuredComment` , see TPS_GST_00381 .

⌋()

**[constr_2530] `InstanceRefs` must be consistent** ⌈ The first `atpContextElement` in the path must be an `atpFeature` of the `atpBase` . For all subsequent `atpContextElement` s, they must be an `atpFeature` of the `atpType` of the previous element (which is an `AtpPrototype` ).

⌋()

**[constr_2531] `AtpInstanceRef` shall be close to the base** ⌈ An `AtpInstanceRef` shall be aggregated such that its relationship to the `AtpClassifier` referenced in the role `atpBase` is unambiguous. This is the case in one of the following situations:

- The `AtpInstanceRef` is aggregated within the `AtpFeature` referenced in the role `atpBase` .
- The `atpBase` is the root of the instance tree. It is the `AtpClassifier` which is aggregating the first `AtpFeature` representing the first (outermost) `atpContextElement` .

⌋()

**[constr_2533] Documentation context is either a feature or an identifiable** ⌈ One particular `DocumentationContext` shall be either a feature or an identifiable but not both at the same time. If this is desired, one should create multiple `DocumentationContext` .

⌋()

**[constr_2534] Limits of unlimited Integer** ⌈ Practically `UnlimitedInteger` shall be limited such that it fits into 64 bit.

If a signed value is represented the min value can be down to -9223372036854775808 (0x80000000000000014) and the max value can be up to 9223372036854775807 ( `0x7fffffffffffffff` ).

If an unsigned value is represented the min value can be down to 0 and the max value can be up to 18446744073709551615 ( `0xffffffffffffffff` ).

⌋()

**[constr_2537] Variation of `PackageableElement` is limited to components resp. modules** ⌈ Variation of `ARElement` in `ARPackage` shall be applied only to elements on a kind of component level. In particular this is `BswModuleDescription` , `Documentation` , `Implementation` , `SwComponentType` , `TimingExtension` . This constraint only applies if the `PackageableElement` is not a blueprint.

⌋()

**[constr_2538] Global reference is limited to certain elements** ⌈ The ability to perform a global reference is limited to `Chapter` , `Topic1` , `Caption` , `Traceable` , `XrefTarget` , `Std` , `Xdoc` , `Xfile`

⌋()

**[constr_2547] Ordered collections cannot be split into partial models** ⌈ Ordered collections cannot be split. In other words: In opposite to unordered collections - which can be distributed between partial models - ordered collections can only be placed as a whole in one of the partial models. Otherwise the merge approach would influence the semantics of the collections.

⌋()

**[constr_2557] No `VariationPoint` s where `vh.latestBindingTime` set to `BlueprintDerivationTime` in system configurations** ⌈ Blueprints are **not** part of a system configuration. In consequence of this, in a system configuration there shall be

no VariationPoint where vh.latestBindingTime is restricted to BlueprintDerivationTime by the meta model.

⌋()

**[constr_2558] If vh.latestBindingTime is BlueprintDerivationTime then there shall only be blueprintCondition / blueprintValue** ⌈ VariationPoint s with vh.latestBindingTime restricted to BlueprintDerivation shall not have swSysCond nor postbuildVariantCondition .

⌋()

**[constr_2559] No nested VariationPoint** ⌈ As blueprintCondition is a DocumentationBlock it could again contain VariationPoint s and therefore would allow nesting of VariationPoint s. This is not intended and shall not be used.

⌋()

**[constr_2567] Undefined Value in Attribute Value Blueprints** ⌈ If a blueprintValue is specified, then the value defined by the AttributeValueVariationPoint is not used and should therefore at least contain one term undefined which is to be refined when deriving objects from this blueprint.

⌋()

**[constr_2572] Unique Control of Document Languages** ⌈ The settings for multiple languages are specified in the top-Level AdminData only

⌋()

**[constr_2573] ICS shall not reference examples** ⌈ ICS is like a productive Model and therefore shall not reference to an EXAMPLE . Such a reference would be useless since the target needs to be ignored in the ICS.

⌋()

**[constr_2574] globalInPackage for global elements only** ⌈ ReferenceBase . globalInPackage is allowed only if isGlobal is set to true.

⌋()

**[constr_2575] blueprintValue in blueprints only** ⌈ blueprintValue is only allowed in blueprints and may not be present in a system description.

⌋()

**[constr_2577] Binding Time in Aggregation Pattern** ⌈ Within VariationPoint , the class ConditionByFormula has an attribute bindingTime which defines the *latest* binding time for this variation point. This binding time is further constrained by the UML tag vh.latestBindingTime that is attached to the aggregation see TPS_GST_00190 , TPS_GST_00220 , TPS_GST_00221 ):

ConditionByFormula . bindingTime $\leq$ *aggregation* . vh.latestBindingTime

⌋()

**[constr_2578] Binding Time in Association Pattern** ⌈ Within `VariationPoint` , the class `ConditionByFormula` has an attribute `bindingTime` which defines the *latest* binding time for this variation point. This binding time is further constrained by the UML tag `vh.latestBindingTime` that is attached to the association (see TPS_GST_00190 , TPS_GST_00220 , TPS_GST_00221 ):

`ConditionByFormula . bindingTime` $\leq$ *association* `. vh.latestBindingTime`

⌋()

**[constr_2579] Binding Time in Attribute Value Pattern** ⌈ The meta class `AttributeValueVariationPoint` has an attribute `bindingTime` which defines the *latest* binding time for this variation point. This binding time is further constrained by the UML tag `vh.latestBindingTime` that is attached to the attribute (see TPS_GST_00190 , TPS_GST_00220 , TPS_GST_00221 ):

`AttributeValueVariationPoint . bindingTime` $\leq$ *attribute* `. vh.latestBindingTime`

⌋()

**[constr_2580] Binding Time in Property Set Pattern** ⌈ The meta class `VariationPoint` has an attribute `bindingTime` which defines the *latest* binding time for this variation point. This binding time is further constrained by the UML tag `vh.latestBindingTime` that is attached to the meta class which is marked as `atpVariation` (see TPS_GST_00190 , TPS_GST_00220 , TPS_GST_00221 ):

`VariationPoint . bindingTime` $\leq$ *meta class* `. vh.latestBindingTime`

⌋()

**[constr_2581] Default life cycle state shall be defined properly** ⌈ `defaultLcState` in `LifeCycleInfoSet` shall reference to a `lcState` defined in the `LifeCycleStateDefinitionGroup` referenced by `usedLifeCycleStateDefinitionGroup` .

⌋()

**[constr_2583] Used life cycle state shall be defined properly** ⌈ `defaultLcState` in `LifeCycleInfo` shall reference to a `lcState` defined in the `LifeCycleStateDefinitionGroup` referenced by `usedLifeCycleStateDefinitionGroup` of the containing `LifeCycleInfoSet` .

⌋()

**[constr_2585] `LifeCycleInfo` shall be unambiguous** ⌈ Within one particular `LifeCycleInfoSet lifeCycleInfo . lcObject` shall be unique. This ensures that the association of a `LifeCycleState` to a `Referrable` is unambiguous.

This contraint applies for a particular point in time under consideration of the period of viability according to TPS_GST_00244 .

⌋()

**[constr_2586] Constraints on `LifeCyclePeriod`** ⌈ The attributes `date` , `arReleaseVersion` , `productRelease` in `LifeCyclePeriod` are mutually exclusive.

⌋()

**[constr_2587] No `System` in `AnyInstanceRef`** ⌈ In consequence of constr_2531 `System` shall not be `contextElement` nor `target` of an `AnyInstanceRef` . Otherwise `atpBase` would not be determined.

⌋()

**[constr_2594] Cyclic value assignments to `SwSystemconst` is not allowed** ⌈ It is explicitly forbidden to assign values to `SwSystemconst` which in turn depend directly or indirectly on this value assignment.

⌋()

**[constr_2595] Footnotes should not be nested** ⌈ Note that even if supported by the meta model, footnotes shall not be nested. Nested footnotes might lead to problems with the processing of the footnote link. In other words `LParagraph` shall not be aggregated with role `ft` within a `LParagraph` which already has the role `ft` .

⌋()

**[constr_2596] Used colors of attributes color and bgcolor** ⌈ The used colors of the attributes color and bgcolor shall base on the 6 digits RGB hex-code following

```
|#([a-fA-F0-9]{6})|
```

.

⌋()

**[constr_2599] Maximum one VariationPoints in `atpMixed`** ⌈ In case an `atpMixed` meta class is aggregated as `atpVariation` there shall not be more than one `VariationPoint` and the `VariationPoint` shall be the last aggregated element.

⌋()

**[constr_2601] Value of `AbstractEnumerationValueVariationPoint`** ⌈ The formula of an `AbstractEnumerationValueVariationPoint` shall evaluate to a value for which a mapping is defined in the `EnumerationMappingTable` which is referenced by the attributes `base` and `enumTable` .

⌋()

**[constr_2602] Completeness of `AnyInstanceRef` referencing `ImplementationDataTypeElement`** ⌈ If the `target` references an `ImplementationDataTypeElement` the `AnyInstanceRef` shall define a `contextElement` reference for

1. each leaf `ImplementationDataTypeElement` in a chain of referencing `ImplementationDataType` s which is not the `target`
2. and each `ImplementationDataTypeElement` of category ARRAY in a chain of referencing `ImplementationDataType` s

Thereby the contexts are created according TPS_GST_00162 from the root to the leaf `ImplementationDataTypeElement` which is either typed (directly or indirectly via `ImplementationDataType` of category TYPE_REFERENCE) or owns the `target` .

⌋()

**[constr_2605] If a SdgClass is referenced then it shall have a caption** ⌈ `destSdg . caption` ==true

⌋()

**[constr_2606] Existence of `upperMultiplicityInfinite` and `upperMultiplicity` of `AbstractMultiplicityRestriction` is mutually exclusive** ⌈ The existence of the elements `upperMultiplicityInfinite` and `upperMultiplicity` of `AbstractMultiplicityRestriction` shall be mutually exclusive.

⌋()

**[constr_2607] `lowerMultiplicity` of `AbstractMultiplicityRestriction` shall be smaller or equal to `upperMultiplicity`** ⌈ `lowerMultiplicity` of `AbstractMultiplicityRestriction` shall be smaller or equal to `upperMultiplicity` .

⌋()

**[constr_4055] ICS may not contain blueprints** ⌈ Since an Implementation Conformance Statement always describes a set of one or more fully configured software modules, a package with category `ICS` it is not allowed to contain sub-packages at any level which have the category `BLUEPRINT` .

⌋()

## 2.16 TPS_SafetyExtensions

**[constr_6200] Safety goals have no decomposed ASIL** ⌈ If a safety requirement is of type `SAFETY_GOAL` the valid values of the `ASIL` attribute are restricted to: `QM, A, B, C,` or `D` .

⌋()

**[constr_6201] Consistency of ASIL values** ⌈ The ASIL of AUTOSAR elements and allocated safety requirements should be *consistent* . An ASIL is consistent if the value at an element is the same or higher of the maximum ASIL of allocated safety requirements.

⌋()

**[constr_6202] Decomposition into two safety requirements** ⌈ A decomposition as specified by TPS_SAFEX_00302 shall be specified at exactly two decomposing safety requirements (not more) for each decomposed requirement.

⌋()

**[constr_6203] Decomposing only one safety requirement** ⌈ Each decomposing requirement specified according to TPS_SAFEX_00302 shall decompose maximum one other requirement.

⌋()

## 2.17 TPS_SoftwareComponentTemplate

**[constr_1000] End-to-end protection is limited to sender/receive communication** ⌈ end-to-end protection applies for sender/receiver communication only

⌋()

**[constr_1001] Value of `dataId` shall be unique** ⌈ The value of the `dataId` shall be unique within the scope of the `System` .

⌋()

**[constr_1004] Mapping of `ApplicationDataType` s in the scope of single `AtomicSwComponentType` s** ⌈ In the scope of `AtomicSwComponentType` . `internalBehavior` . `dataTypeMapping` , each `ApplicationDataType` shall be mapped to exactly one `ImplementationDataType` .

⌋()

**[constr_1005] Compatibility of `ImplementationDataType` s mapped to the same `ApplicationDataType`** ⌈ It is required that `ImplementationDataType` s which are taken for connecting corresponding elements of `PortInterface` s and thus refer to compatible `ApplicationDataType` s are also compatible among each other (so that RTE is able to cope with possible connections by converting the data accordingly).

⌋()

**[constr_1006] applicable data categories** ⌈ Table `table_3a_CategoriesOverview` defines the applicable `category` s depending on specific model elements related to data definition properties.

⌋()

**[constr_1007] Allowed attributes of `SwDataDefProps` for `ApplicationDataType` s** ⌈ The allowed attributes of `SwDataDefProps` for `Application-`

DataType s and their allowed multiplicities are listed as an overview in table table_3a_CategoriesAppl .

⌋()

**[constr_1008] Applicability of category s STRUCTURE and ARRAY** ⌈ The categories STRUCTURE and ARRAY correspond to ApplicationCompositeDataType s whereas all other category s can be applied only for ApplicationPrimitiveDataType s.

⌋()

**[constr_1009] SwDataDefProps applicable to ImplementationDataType s** ⌈ A complete list of the SwDataDefProps and other attributes and their multiplicities which are allowed for a given category is shown in table table_3a_CategoriesImpl .

⌋()

**[constr_1010] If nativeDeclaration does not exist** ⌈ If nativeDeclaration does not exist in the SwBaseType it is required that the shortName (e.g. “uint8”) of the corresponding ImplementationDataType is equal to a name of one of the Platform or Standard Types predefined in AUTOSAR code.

⌋()

**[constr_1011] category of SwBaseType** ⌈ For the attribute SwBaseType . category only the values FIXED_LENGTH and VOID are supported.

⌋()

**[constr_1012] Value of category is FIXED_LENGTH** ⌈ If the value of the attribute SwBaseType . category is set to FIXED_LENGTH then the attribute baseTypeSize shall be filled with content.

⌋()

**[constr_1014] Supported value encodings for SwBaseType** ⌈ The supported values for attribute BaseTypeDirectDefinition . baseTypeEncoding are:

- 1C : One’s complement
- 2C : Two’s complement
- BCD-P : Packed Binary Coded Decimals
- BCD-UP : Unpacked Binary Coded Decimals
- DSP-FRACTIONAL : Digital Signal Processor
- SM : Sign Magnitude
- IEEE754 : floating point numbers
- ISO-8859-1 : single-byte coded character
- ISO-8859-2 : single-byte coded character

- WINDOWS-1252 : single-byte coded character
- UTF-8 : UCS Transformation Format 8
- UTF-16 : Character encoding for Unicode *code points* based on 16 bit *code units* [6]
- UCS-2 : Universal Character Set 2
- NONE : Unsigned Integer
- VOID : corresponds to a void in C. The encoding is not formally specified here.
- BOOLEAN : This represents an unsigned integer to be interpreted as boolean. The value shall be interpreted as true if the value of the unsigned integer is 1 and it shall be interpreted as false if the value of the unsigned integer is 0.

  A CompuMethod shall be referenced by the corresponding AutosarDataType that implements the common sense behind the boolean concept, i.e. define a TEXTTABLE with two CompuScale s: e.g. true –> 1, false –> 0.

⌋()

**[constr_1015] Prioritization of SwDataDefProps** ⌈ The prioritization and usage of attributes of meta-class SwDataDefProps shall follow the restrictions given in table table_3a_DataDefPropsUsageDetails .

⌋()

**[constr_1016] Restriction of invalidValue for ImplementationDataType and ImplementationDataTypeElement** ⌈ invalidValue for ImplementationDataType and ImplementationDataTypeElement is restricted to to be either a compatible NumericalValueSpecification , TextValueSpecification (caution, constr_1284 applies) or a ConstantReference that in turn points to a compatible ValueSpecification .

⌋()

**[constr_1017] Supported combinations of swImplPolicy and swCalibrationAccess** ⌈ The table tab_3a_Supported_20_combinations_20_of_20_SwImplPolicy_20_and_20_SwCalibr defines the supported combinations of swImplPolicy and swCalibrationAccess attribute setting.

⌋()

**[constr_1018] measurementPoint shall not be referenced by a VariableAccess aggregated by RunnableEntity in the role dataReadAccess** ⌈ Due to the nature of dataElement s characterized by setting the swImplPolicy to measurementPoint , such dataElement s shall not be referenced by a VariableAccess aggregated by RunnableEntity in the role dataReadAccess .

⌋()

**[constr_1020] `ParameterDataPrototype` needs to be of compatible data type as referenced in `sharedAxisType`** ⌈ Finally, the `ParameterDataPrototype` assigned in `swCalprmRef` shall be typed by data type compatible to `sharedAxisType` .

⌋()

**[constr_1022] Limits shall be defined for each direction of `CompuMethod`** ⌈ In case that both domains are specified in the `CompuMethod` both shall have explicitly defined limits.

⌋()

**[constr_1024] Stepwise definition of `CompuMethod` s** ⌈ In a bound model, the intervals (i.e. determined by attributes `CompuScale` . `lowerLimit` and `CompuScale` . `upperLimit` ) defined by `CompuScale` s used in the context of a given `CompuMethod` shall **not** overlap.

This applies **for all** possible values of `CompuMethod` . `category` .

⌋()

**[constr_1025] Avoid division by zero in rational formula** ⌈ The rational formula shall not yield any division by zero.

⌋()

**[constr_1026] Compatibility of `Unit` s** ⌈ For data types or prototypes, units should be referenced from within the associated `CompuMethod` . But if it is referenced from within `SwDataDefProps` and/or `PhysConstrs` (for exceptional use cases) it shall be compatible (for more details please refer to constr_1052 ) to the ones referenced from the referred `CompuMethod` .

⌋()

**[constr_1027] Types for record layouts** ⌈ Because `ParameterDataPrototype` s have a `isOfType` -relation to `ApplicationDataType` s or `ImplementationDataType` s the related data types shall properly match to the details as specified in `swDataDefProps` .

⌋()

**[constr_1029] `ConstantSpecificationMapping` and `ConstantSpecification`** ⌈ It is required that one `ConstantSpecification` referenced from a `ConstantSpecificationMapping` needs to be defined in the application domain ( `applConstant` ) and the other referenced `ConstantSpecification` needs to be defined in the implementation domain ( `implConstant` ).

⌋()

**[constr_1030] `ParameterSwComponentType` references `ConstantSpecificationMappingSet`** ⌈ `ParameterSwComponentType` : here the `ConstantSpeci-`

ficationMappingSet is directly associated by the ParameterSwComponentType .

⌋()

**[constr_1031] NvBlockSwComponentType references ConstantSpecificationMappingSet** ⌈ NvBlockSwComponentType : in this case the ConstantSpecificationMappingSet is associated with the aggregated NvBlockDescriptor .

⌋()

**[constr_1032] DelegationSwConnector can only connect PortPrototype s of the same kind** ⌈ A DelegationSwConnector can only connect PortPrototype s of the same kind, i.e. PPortPrototype to PPortPrototype and RPortPrototype to RPortPrototype .

⌋()

**[constr_1033] Communication scenarios for sender/receiver communication** ⌈ For sender/receiver communication, it is not allowed to create a communication scenario where n sender are connected to m receivers where m and n are **both** greater than 1.

⌋()

**[constr_1035] Recursive definition of CompositionSwComponentType** ⌈ The recursive definition of a CompositionSwComponentType that eventually contains a SwComponentPrototype typed by the same CompositionSwComponentType shall not be feasible.

⌋()

**[constr_1036] Connect kinds of PortInterface s** ⌈ It shall not be possible to connect PortPrototype s typed by PortInterface s of different kinds. Subclasses of DataInterface make an exception from this rule and can be used for creating connections to each other.

⌋()

**[constr_1037] Client shall not be connected to multiple servers** ⌈ A client shall not be connected to multiple servers such that an operation call would be handled by more than one server.

⌋()

**[constr_1038] Reference to ApplicationError** ⌈ A possibleError referenced by a ClientServerOperation shall be owned by the PortInterface that also owns the ClientServerOperation .

⌋()

**[constr_1039] Relevance of `swImplPolicy`** ⌈ It is not possible to define a mapping between an element where the `swImplPolicy` is set to `queued` and an other element where the `swImplPolicy` is set differently.

⌋()

**[constr_1040] Conversion of `SenderReceiverInterface` s** ⌈ The conversion of elements of `SenderReceiverInterface` s is possible if one of the following conditions applies:

- The `AutosarDataType` s of the referred `DataPrototype` s are compatible as described in chapter `chap_3a_Compatibility_of_Data_Types` .
- A conversion of the data as described in chapter `chap_3a_Data_20_Conversion` is available.
- A `DataPrototypeMapping` . `firstToSecondDataTransformation` is defined.

⌋()

**[constr_1041] Conversion of `ClientServerInterface` s** ⌈ Either the `AutosarDataType` s of the referred `ArgumentDataPrototype` s are compatible as described in chapter `chap_3a_Compatibility_of_Data_Types` or a conversion of the data as described in chapter `chap_3a_Data_20_Conversion` is available.

⌋()

**[constr_1043] `PortInterface` vs. `ComSpec`** ⌈ The allowed combinations of a specific kind of `PortInterface` and a kind of `ComSpec` are documented in Table `table_3a_Port_Interface_vs_Com_Spec` .

⌋()

**[constr_1044] Applicability of `DataFilter`** ⌈ According to the origin of `DataFilter` , i.e. ISO 17356-4 specification [7] , `DataFilter` s can only be applied to values with an integer base type.

⌋()

**[constr_1045] Supported value encodings for `SwBaseType` in the context of `PortInterface` s** ⌈ The supported value encodings for the usage within a `PortInterface` are:

- `2C` : Two's complement
- `IEEE754` : floating point numbers
- `ISO-8859-1` : single-byte coded character
- `ISO-8859-2` : single-byte coded character
- `WINDOWS-1252` : single-byte coded character
- `UTF-8` : UCS Transformation Format 8

- UTF-16 : Character encoding for Unicode *code points* based on 16 bit *code units* [6]
- UCS-2 : Universal Character Set 2
- NONE : Unsigned Integer
- BOOLEAN : This represents an integer to be interpreted as boolean.

⌋()

**[constr_1046] Applicability of [constr_1045]** ⌈ constr_1045 applies **only** if the value of the attribute isService is set to false .

⌋()

**[constr_1047] Compatibility of ApplicationPrimitiveDataType s** ⌈ Instances of ApplicationPrimitiveDataType are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
   (a) They have the same category (see table in figure table_3a_CategoriesAppl ).
   (b) The swDataDefProps attached to the M1 data types are compatible. The meaning of this statement is explained in section chap_3a_Compatibility_of_SwDataDefProps .
2. In the context of using the ApplicationPrimitiveDataType , a DataPrototypeMapping exists that refers to a DataPrototype typed by one of the ApplicationPrimitiveDataType s in the role firstDataPrototype and to another DataPrototype typed by the other ApplicationPrimitiveDataType in the role secondDataPrototype .
3. In the context of using the ApplicationPrimitiveDataType , a DataPrototypeMapping exists that refers to a DataPrototype typed by the ApplicationPrimitiveDataType in the role secondDataPrototype and to another DataPrototype typed by an ApplicationCompositeDataType in the role firstDataPrototype and additionally for the side of the ApplicationCompositeDataType a corresponding ApplicationCompositeDataTypeSubElementRef exists in the role firstElement that in turn references an ApplicationCompositeElementDataPrototype .

⌋()

**[constr_1048] Compatibility of ApplicationRecordDataType s** ⌈ Instances of ApplicationRecordDataType s are compatible if and only if one of the following conditions applies:

1. All element s *at the same record position* are of compatible AutosarDataType s either ApplicationCompositeDataType s or ApplicationPrimitiveDataType s).

2. In the context of a DataPrototypeMapping , for each ApplicationRecordElement of the required ApplicationRecordDataType a SubElementMapping exists such that a ApplicationCompositeDataTypeSubElementRef in the role firstElement or secondElement exists that references the required ApplicationRecordElement **and** a corresponding ApplicationCompositeDataTypeSubElementRef exists in the **other** role (i.e. secondElement or firstElement ) that in turn references an ApplicationRecordElement of the provided ApplicationRecordDataType .

⌋()

**[constr_1049] Compatibility of ApplicationArrayDataType s** ⌈ Instances of ApplicationArrayDataType are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
   (a) Their element s are of a compatible AutosarDataType s (either ApplicationCompositeDataType s or ApplicationPrimitiveDataType s).
   (b) The attributes maxNumberOfElements and arraySizeSemantics (given the existence) have identical values.
2. In the context of a DataPrototypeMapping , for the ApplicationArrayElement of the required ApplicationArrayDataType a SubElementMapping exists such that a ApplicationCompositeDataTypeSubElementRef in the role firstElement or secondElement exists that references the required ApplicationArrayElement **and** a corresponding ApplicationCompositeDataTypeSubElementRef exists in the **other** role (i.e. secondElement or firstElement ) that in turn references an ApplicationArrayElement of the provided ApplicationArrayDataType .

⌋()

**[constr_1050] Compatibility of ImplementationDataType s** ⌈ Instances of ImplementationDataType are compatible if and only if after all type-references are resolved one of the following rules apply:

1. All of the following subconditions apply:
   (a) They have the same category (see table table_3a_CategoriesImpl )
   (b) They have the identical structure (this refers to ImplementationDataTypeElement and their subElement s).
   (c) The attributes arraySize and arraySizeSemantics have (given the existence) identical values.
   (d) The swDataDefProps attached to the M1 data types are compatible. The meaning of this statement is explained in section chap_3a_Compatibility_of_SwDataDefProps .

2. In the context of using the ImplementationDataType , a DataPrototypeMapping exists that refers to a DataPrototype typed by one of the ImplementationDataType s in the role firstDataPrototype and to another DataPrototype typed by the other ImplementationDataType in the role secondDataPrototype .
3. In the context of using the ImplementationDataType , a DataPrototypeMapping exists that refers to a DataPrototype typed by the ImplementationDataType s in the role secondDataPrototype and to another DataPrototype typed by an ImplementationDataType with a subElement in the role firstDataPrototype and additionally for the side of the ImplementationDataType with a subElement a corresponding ImplementationDataTypeSubElementRef exists in the role firstElement that in turn references an ImplementationDataTypeElement .

⌋()

**[constr_1051] Compatibility of SwDataDefProps** ⌈ SwDataDefProps are compatible if and only if:

1. They refer to compatible Unit definitions, or neither of them has an associated Unit .
2. They refer to compatible conversion methods (see chapter chap_3a_CompatibilityOfCompuMethods ) or neither of them associates such a method.
3. One of the following conditions apply to ValueSpecification s aggregated in the role invalidValue for being considered compatible (after following and resolving indirections created by ConstantReference ):
   (a) both are ApplicationValueSpecification s and the values are compatible according to [TPS_GST_02501] .
   (b) both are NumericalValueSpecification s and the values are compatible according to [TPS_GST_02501] .
   (c) both are TextValueSpecification s and the values are identical.
   (d) both are ArrayValueSpecification s and the values are identical.
   (e) both are RecordValueSpecification s and the values are identical.
   (f) if one is a NumericalValueSpecification and the other one is an ApplicationValueSpecification then the check for compatibility shall apply the CompuMethod on the physical value such that a comparison on the implementation level becomes possible. [TPS_GST_02501] applies if one is a NumericalValueSpecification and the other one is an ApplicationValueSpecification and the application of the CompuMethod on the side of the ApplicationValueSpecification does not yield a valid number a comparison is not possible. .

4. They refer to compatible data constraints `dataConstr` .
5. They refer to compatible `swRecordLayout` s

All other attributes (e.g. `swCalibrationAccess` do not affect compatibility).

⌋()

**[constr_1052] Compatibility of `Unit` s** ⌈ Two `Unit` definitions are compatible if and only if:

1. They have compatible (see [TPS_GST_02501] ) values of attributes `factorSiToUnit` and `offsetSiToUnit` .
2. They either refer to identical definitions of `PhysicalDimension` or neither of them associates a `PhysicalDimension` .

⌋()

**[constr_1053] Compatibility of `PhysicalDimension` s** ⌈ Two `PhysicalDimension` definitions are compatible if and only if the values of

- `lengthExp`
- `massExp`
- `timeExp`
- `currentExp`
- `temperatureExp`
- `molarAmountExp`
- `luminousIntensityExp`

are identical and **either** the `shortName` s are identical **or** a `PhysicalDimensionMapping` exists that maps one of the `PhysicalDimension` s in the role `firstPhysicalDimension` and the other `PhysicalDimension` in the role `secondPhysicalDimension` .

⌋()

**[constr_1054] No `DataConstr` available at the provider** ⌈ If the provider defines no constraints it is only compatible with a receiver which also defines no constraints at all.

⌋()

**[constr_1055] `ImplementationDataType` has `category` `VALUE`** ⌈ The attributes `baseType` shall refer to a compatible `SwBaseType`

⌋()

**[constr_1056]** **ImplementationDataType has category TYPE_REFERENCE** ⌈ The ImplementationDataType s referenced by the attributes SwDataDefProps . implementationDataType shall be compatible .

⌋()

**[constr_1057]** **ImplementationDataType has category DATA_REFERENCE** ⌈ The attributes SwDataDefProps . swPointerTargetProps shall have identical targetCategory and shall refer to SwDataDefProps where all attributes are identical

⌋()

**[constr_1058]** **ImplementationDataType has category FUNCTION_REFERENCE** ⌈ The attributes SwDataDefProps . swPointerTargetProps . functionPointerSignature shall refer to BswModuleEntry s which each resolve to the **same function signature** .

⌋()

**[constr_1059]** **Compatibility of data types with category VALUE** ⌈ An ApplicationDataType of category VALUE can only be mapped/connected to an ImplementationDataType which also has category VALUE .

⌋()

**[constr_1060]** **Compatibility of data types with category ARRAY , VAL_BLK** ⌈ An ApplicationDataType of category ARRAY , VAL_BLK can only be mapped/connected to

- an ImplementationDataType of category ARRAY **or**
- an ImplementationDataType that represents a Variable-Size Array Data Type (see TPS_SWCT_01610 ).

⌋()

**[constr_1061]** **Compatibility of data types with category STRUCTURE** ⌈ An ApplicationDataType of category STRUCTURE can only be mapped/connected to an ImplementationDataType of category STRUCTURE .

⌋()

**[constr_1063]** **Compatibility of data types with category BOOLEAN** ⌈ An ApplicationDataType of category BOOLEAN can only be mapped/connected to an ImplementationDataType of category VALUE .

⌋()

**[constr_1064]** **Compatibility of data types with category COM_AXIS , RES_AXIS , CURVE , MAP , CUBOID , CUBE_4 , or CUBE_5** ⌈ An ApplicationDataType of category COM_AXIS , RES_AXIS , CURVE , MAP , CUBOID , CUBE_4 , or CUBE_5 can only

be mapped/connected to an ImplementationDataType of category STRUCTURE or ARRAY .

⌋()

**[constr_1066] Forbidden mappings to ImplementationDataType** ⌈ An ApplicationDataType shall never be mapped to an ImplementationDataType of of category UNION , DATA_REFERENCE , or FUNCTION_REFERENCE .

⌋()

**[constr_1068] Compatibility of VariableDataPrototype s or ParameterDataPrototype s typed by primitive data types** ⌈ Two VariableDataPrototype s or ParameterDataPrototype s of ApplicationPrimitiveDataType s or ImplementationDataType s of category VALUE , BOOLEAN , or STRING are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
   (a) They are typed by (read “refer to”) compatible AutosarDataType s
   (b) The two VariableDataPrototype s or ParameterDataPrototype s have identical shortName s. This is required to map VariableDataPrototype s in unordered SenderReceiverInterface s, NvDataInterface s and ParameterInterface s.
   (c) The attribute swImplPolicy is either set to queued for both or none of the VariableDataPrototype s.
2. In the context of a DataPrototypeMapping , one of the applicable VariableDataPrototype s or ParameterDataPrototype s is referenced by the DataPrototypeMapping in the role firstDataPrototype and the other VariableDataPrototype s or ParameterDataPrototype s is referenced by the same DataPrototypeMapping in the role secondDataPrototype .

⌋()

**[constr_1069] Compatibility of PortPrototype s of different DataInterface s in the context of AssemblySwConnector s** ⌈ PortPrototype s of different DataInterface s are compatible if and only if

1. One of the following conditions applies:
   (a) For each VariableDataPrototype or ParameterDataPrototype defined in the context of the DataInterface of the required PortPrototype a compatible (see constr_1068 ) VariableDataPrototype or ParameterDataPrototype exists in the DataInterface of the provided PortPrototype .

   The shortName s of VariableDataPrototype s and ParameterDataPrototype s are used to identify the pair.

(b) A VariableAndParameterInterfaceMapping . dataMapping exists for which the following conditions apply:

i. It is referenced by the corresponding SwConnector .

ii. It references one of the two VariableDataPrototype s or ParameterDataPrototype s in the role firstDataPrototype and the other in the role secondDataPrototype .

2. For each such pair, the values of their isService attributes are identical.

⌋()

**[constr_1070] Compatibility of PortPrototype s of different DataInterface s in the context of DelegationSwConnector s** ⌈ PortPrototype s of different DataInterface s are compatible if and only if

1. One of the following conditions applies:

(a) For each VariableDataPrototype or ParameterDataPrototype defined in the context of the DataInterface of the required inner PortPrototype a compatible VariableDataPrototype or ParameterDataPrototype exists in the DataInterface of the required outer PortPrototype .

The shortName of VariableDataPrototype s and ParameterDataPrototype s are used to identify the pair.

constr_1071 defines which PortInterface elements are compatible depending on the PortInterface type and the swImplPolicy attributes of the PortInterface elements.

(b) A VariableAndParameterInterfaceMapping . dataMapping exists for which the following conditions apply:

i. It is referenced by the corresponding SwConnector .

ii. It references one of the two VariableDataPrototype s or ParameterDataPrototype s in the role firstDataPrototype and the other in the role secondDataPrototype .

2. One of the following conditions applies:

(a) For at least one VariableDataPrototype or ParameterDataPrototype defined in the context of the SenderReceiverInterface , NvDataInterface or ParameterInterface of the provided inner PortPrototype a compatible VariableDataPrototype or ParameterDataPrototype exists in the SenderReceiverInterface , NvDataInterface or ParameterInterface of the provided outer PortPrototype .

The shortName s of VariableDataPrototype s and ParameterDataPrototype s are used to identify the pair.

constr_1071 defines which PortInterface elements are compatible depending on the PortInterface type and the swImplPolicy attributes of the PortInterface elements.

(b) A VariableAndParameterInterfaceMapping . dataMapping exists for which the following conditions apply:

i. It is (if a corresponding SwConnector already exists) referenced by the corresponding SwConnector .

ii. It references one of the two VariableDataPrototype s or ParameterDataPrototype s in the role firstDataPrototype and the other in the role secondDataPrototype .

3. For each such pair, the values of their isService attributes are identical.

⌋()

**[constr_1071] compatibility of ParameterDataPrototype and VariableDataPrototype** ⌈ Combinations of ParameterDataPrototype and VariableDataPrototype used in PortPrototype s typed by various kinds of PortInterface s shall only be allowed where Table tab_3a_Overview_20_of_20_compatibility_20_of_20_ParameterDataPrototype_20 contains the value “yes”.

⌋()

**[constr_1072] Compatibility of ModeSwitchInterface s in the context of an AssemblySwConnector** ⌈ PortPrototype s of different ModeSwitchInterface s are compatible if and only if

1. One of the following conditions applies:

(a) For the ModeDeclarationGroupPrototype defined in the context of the ModeSwitchInterface of the required PortPrototype a compatible ModeDeclarationGroupPrototype exists in the ModeSwitchInterface of the provided PortPrototype .

(b) A ModeInterfaceMapping . modeMapping exists for which the following conditions apply:

i. It is referenced by the corresponding SwConnector .

ii. It references one of the two ModeDeclarationGroupPrototype s in the role firstModeGroup and the other in the role secondModeGroup .

2. For each such pair, the values of their isService attributes are identical.

⌋()

**[constr_1073] Compatibility of `ModeSwitchInterface` s in the context of an `DelegationSwConnector`** ⌈ `PortPrototype` s of different `ModeSwitchInterface` s are compatible if and only if

1. One of the following conditions applies:
   (a) For the `ModeDeclarationGroupPrototype` defined in the context of the `ModeSwitchInterface` of the inner `PortPrototype` a compatible `ModeDeclarationGroupPrototype` exists in the `ModeSwitchInterface` of the outer `PortPrototype` .
   (b) A `ModeInterfaceMapping` . `modeMapping` exists for which the following conditions apply:
      i. It is referenced by the corresponding `SwConnector` .
      ii. It references one of the two `ModeDeclarationGroupPrototype` s in the role `firstModeGroup` and the other in the role `secondModeGroup` .
2. For each such pair, the values of their `isService` attributes are identical.

⌋()

**[constr_1074] Compatibility of `ModeDeclarationGroupPrototype` s** ⌈ `ModeDeclarationGroupPrototype` s are compatible if and only if one of the following conditions applies:

1. They are typed by (read “refer to”) compatible `ModeDeclarationGroup` s.
2. A `ModeDeclarationGroupPrototypeMapping` exists that identifies the differently named `ModeDeclarationGroupPrototype` s that correlate with each other. constr_1210 applies.

⌋()

**[constr_1075] Compatibility of `ModeDeclarationGroup` s** ⌈ `ModeDeclarationGroup` s are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
   (a) They define an identical number of `ModeDeclaration` s.
   (b) Each `ModeDeclaration` on the required side corresponds to a `ModeDeclaration` on the provided side with an identical `shortName` .
   (c) The `initialMode` s on both sides refer to `ModeDeclaration` s with identical `shortName` s.
   (d) The attribute `ModeDeclarationGroup` . `modeUserErrorBehavior` . `errorReactionPolicy` has identical values on both sides.
   (e) The attribute `ModeDeclarationGroup` . `modeManagerErrorBehavior` . `errorReactionPolicy` has identical values on both sides.

(f) The attribute ModeDeclarationGroup . modeUserErrorBehavior . defaultMode either does not exist on both sides or refers on both sides to ModeDeclaration s with identical shortName s.

(g) The attribute ModeDeclarationGroup . modeManagerErrorBehavior . defaultMode either does not exist on both sides or refers on both sides to ModeDeclaration s with identical shortName s.

(h) one of the following subconditions applies:

- the attribute category has the value ALPHABETIC_ORDER on both sides.
- the attribute category has the value EXPLICIT_ORDER on both sides **and** the matching ModeDeclaration s according to 1(b) have the identical values of the attributes ModeDeclaration . value **and** also the value of ModeDeclarationGroup . onTransitionValue matches on both sides.

2. A ModeDeclarationMapping is applied which identifies the corresponding ModeDeclaration s.

In addition, the compatibility of corresponding ModeTransition s shall be checked, i.e. constr_1194 and constr_1245 apply.

⌋()

**[constr_1076] Compatibility of ArgumentDataPrototype s** ⌈ Two ArgumentDataPrototype s are compatible if and only if

1. They are typed by compatible AutosarDataType s **or** a ClientServerOperationMapping . argumentMapping exists that references one ArgumentDataPrototype in the role firstDataPrototype and the other ArgumentDataPrototype in the role secondDataPrototype .
2. They have the same value of the argument direction ( in , out or inout ), i.e. constr_1268 applies.

⌋()

**[constr_1077] Compatibility of ApplicationError s** ⌈ Two ApplicationError s are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
   (a) They have the same shortName .
   (b) They have the same attributes. Especially the errorCode shall be identical in both ApplicationError s.
2. A ClientServerInterfaceMapping . errorMapping exists that references one of the ApplicationError s in the role firstApplicationError and the other ApplicationError s in the role secondApplicationError .

⌋()

**[constr_1078] Compatibility of `ClientServerOperation` s** ⌈ Two `ClientServerOperation` s are compatible if their signatures match. In particular, they are compatible if and only if

1. They have the same number of `ArgumentDataPrototype` s.
2. The n-th arguments of both `ClientServerOperation` s are compatible. This implies ordering of `ArgumentDataPrototype` s.
3. They have the same `shortName` (again allows for mapping in `PortInterface` s).
4. The required `ClientServerOperation` specifies a compatible `ApplicationError` for each `ApplicationError` that is possibly raised by the provided `ClientServerOperation` , maybe more. Thereby, `ClientServerOperation` s that refer to a `possibleError` that represents the value `E_OK` are compatible to `ClientServerOperation` s that do refer to `possibleError` s where none of them represents the value `E_OK` .

⌋()

**[constr_1079] Compatibility of `ClientServerInterface` s in the context of an `AssemblySwConnector`** ⌈ `ClientServerInterface` s are compatible if and only if

1. One of the following conditions applies:
   (a) For each `ClientServerOperation` defined in the context of the `ClientServerInterface` of the required `PortPrototype` a compatible `ClientServerOperation` exists in the `ClientServerInterface` of the provided `PortPrototype` . The `shortName` s of `ClientServerOperation` s are used to identify the pair.
   (b) A `ClientServerInterfaceMapping` . `operationMapping` exists for which the following conditions apply:
      i. It is referenced by the corresponding `SwConnector` .
      ii. It references one of the two `ClientServerOperation` s in the role `firstOperation` and the other in the role `secondOperation` .
2. For each such pair, the values of their `isService` attributes are identical.

⌋()

**[constr_1080] Compatibility of `ClientServerInterface` s in the context of an `DelegationSwConnector`** ⌈ `ClientServerInterface` s are compatible if and only if

1. One of the following conditions applies:

(a) For each ClientServerOperation defined in the context of the ClientServerInterface of the required inner PortPrototype a compatible ClientServerOperation exists in the ClientServerInterface of the required outer PortPrototype . The shortName s of ClientServerOperation s are used to identify the pair.

(b) A ClientServerInterfaceMapping . operationMapping exists for which the following conditions apply:

i. It is referenced by the corresponding SwConnector .

ii. It references one of the two ClientServerOperation s in the role firstOperation and the other in the role secondOperation .

2. One of the following conditions applies:

(a) For at least one ClientServerOperation defined in the context of the ClientServerInterface of the provided inner PortPrototype a compatible ClientServerOperation exists in the ClientServerInterface of the provided outer PortPrototype . The shortName s of ClientServerOperation s are used to identify the pair.

(b) A ClientServerInterfaceMapping . operationMapping exists for which the following conditions apply:

i. It is referenced by the corresponding SwConnector .

ii. It references one of the two ClientServerOperation s in the role firstOperation and the other in the role secondOperation .

3. For each such pair, the values of their isService attributes are identical.

⌋()

**[constr_1081] Compatibility of TriggerInterface s in the context of an AssemblySwConnector** ⌈ TriggerInterface s are compatible if and only if

1. One of the following conditions applies:

(a) For each Trigger defined in the context of the TriggerInterface of the required PortPrototype a compatible Trigger exists in the TriggerInterface of the provided PortPrototype . The shortName s of Trigger are used to identify the pair.

(b) A TriggerInterfaceMapping . triggerMapping exists for which the following conditions apply:

i. It is referenced by the corresponding SwConnector .

ii. It references one of the two Trigger s in the role firstTrigger and the other in the role secondTrigger .

2. For each such pair, the values of their isService attributes are identical.

⌋()

**[constr_1082] Compatibility of `TriggerInterface` s in the context of an `DelegationSwConnector`** ⌈ `TriggerInterface` s are compatible if and only if all of the following conditions apply:

1. One of the following subconditions applies:
   (a) For each `Trigger` defined in the context of the `TriggerInterface` of the **required** inner `PortPrototype` a compatible `Trigger` exists in the `TriggerInterface` of the **required** outer `PortPrototype` . The `shortName` s of `Trigger` are used to identify the pair.
   (b) For at least one `Trigger` defined in the context of the `TriggerInterface` of the **provided** outer `PortPrototype` a compatible `Trigger` exists in the `TriggerInterface` of the **provided** inner `PortPrototype` . The `shortName` s of `Trigger` are used to identify the pair.
   (c) A `TriggerInterfaceMapping` . `triggerMapping` exists for which all of the following conditions apply:
      i. It is referenced by the corresponding `SwConnector` .
      ii. It references one of the two `Trigger` s in the role `firstTrigger` and the other in the role `secondTrigger` .
2. For each such pair, the values of their `isService` attributes are identical.

⌋()

**[constr_1083] Compatibility of `Trigger` s** ⌈ `Trigger` s are compatible if they have an identical `shortName` .

⌋()

**[constr_1084] delegation of a provided outer `PortPrototype`** ⌈ The delegation of a provided outer `PortPrototype` is properly defined if the following criteria are fulfilled:

1. For each `VariableDataPrototype` or `ParameterDataPrototype` present in the `SenderReceiverInterface` , `NvDataInterface` , or `ParameterInterface` of the provided outer `PortPrototype` at least one connection via `DelegationSwConnector` to a provided inner `PortPrototype` **or** `PassThroughSwConnector` to a required outer `PortPrototype` with a compatible `VariableDataPrototype` or `ParameterDataPrototype` in the `SenderReceiverInterface` `NvDataInterface` or `ParameterInterface` of the provided inner `PortPrototype` **or** required outer `PortPrototype` exists. Either the `shortName` s of `VariableDataPrototype` s or `ParameterDataPrototype` s are used to identify the pair or a `PortInterfaceMapping` defines which differently named `PortInterface` elements correlate with each other. Table `tab_3a_Overview_20_of_20_compatibility_20_of_20_ParameterDataPrototyp`

defines which PortInterface elements are compatible depending on the kind of PortInterface and the swImplPolicy attributes of the PortInterface elements.

2. For each VariableDataPrototype provided by a PRPortPrototype that is typed by a SenderReceiverInterface or NvDataInterface and that is referenced in the role outerPort by a DelegationSwConnector a corresponding VariableDataPrototype owned by an innerPort shall be provided by either a PPortPrototype or a PRPortPrototype . Either the shortName s of VariableDataPrototype s are used to identify the pair or a PortInterfaceMapping defines which differently named PortInterface elements correlate with each other.

3. For the ModeDeclarationGroupPrototype present in the ModeSwitchInterface of the provided outer PortPrototype exactly one connection via DelegationSwConnector to a provided inner PortPrototype **or** PassThroughSwConnector to a required outer PortPrototype with a compatible ModeDeclarationGroupPrototype in the ModeSwitchInterface of the provided inner PortPrototype **or** required outer PortPrototype exists. Either the shortName s of ModeDeclarationGroupPrototype s are used to identify the pair or a PortInterfaceMapping defines which differently named PortInterface elements correlate with each other.

4. For each ClientServerOperation present in the ClientServerInterface of the provided outer PortPrototype exactly one connection via DelegationSwConnector to a provided inner PortPrototype **or** PassThroughSwConnector to a required outer PortPrototype with a compatible ClientServerOperation in the ClientServerInterface of the provided inner PortPrototype **or** required outer PortPrototype exists. Either the shortName s of ClientServerOperation s are used to identify the pair or a PortInterfaceMapping defines which differently named PortInterface elements correlate with each other.

5. For each Trigger present in the TriggerInterface of the provided outer PortPrototype exactly one connection via DelegationSwConnector to a provided inner PortPrototype **or** PassThroughSwConnector to a required outer PortPrototype with a compatible Trigger in the TriggerInterface of the provided inner PortPrototype **or** required outer PortPrototype exists. Either the shortName s of Trigger s are used to identify the pair or a PortInterfaceMapping defines which differently named PortInterface elements correlate with each other.

⌋()

**[constr_1085] Compatibility in the case of a flat ECU extract** ⌈ PortPrototype s of different SenderReceiverInterface s, NvDataInterface s, and ParameterInterface s are compatible if and only if for at least one VariableDataPrototype or ParameterDataPrototype defined in the context of the SenderReceiverInterface , NvDataInterface , or ParameterInterface of

the RPortPrototype a compatible VariableDataPrototype or ParameterDataPrototype exists in the SenderReceiverInterface , NvDataInterface , or ParameterInterface of the provided PortPrototype . The compatibility of PortInterface elements depends on the kind of PortInterface and the swImplPolicy attributes of the PortInterface elements.

Either the shortName s of VariableDataPrototype s and ParameterDataPrototype s are used to identify the pair or a PortInterfaceMapping defines which differently named PortInterface elements correlate with each other.

⌋()

**[constr_1086] SwConnector between two specific PortPrototype s** ⌈ Each pair of PortPrototype s can only be connected by one and only one SwConnector .

⌋()

**[constr_1087] AssemblySwConnector inside CompositionSwComponentType** ⌈ An AssemblySwConnector can only connect PortPrototype s of SwComponentPrototype s that are owned by the same CompositionSwComponentType

⌋()

**[constr_1088] DelegationSwConnector inside CompositionSwComponentType** ⌈ A DelegationSwConnector can only connect a PortPrototype of a SwComponentPrototype that is owned by the same CompositionSwComponentType that also owns the connected delegation PortPrototype .

⌋()

**[constr_1090] WaitPoint and RunnableEntity** ⌈ A single RunnableEntity can actually wait only at a single WaitPoint provided that the RunnableEntity can only be scheduled a single time This constraint is valid at least in the ISO 17356-3 [8] standard where an extended task (that can have wait points) can only exist a single time in the context of the scheduler. .

⌋()

**[constr_1091] RTEEvent s that can unblock a WaitPoint** ⌈ The only RTEEvent s that are qualified for unblocking a WaitPoint are:

- DataReceivedEvent
- DataSendCompletedEvent
- ModeSwitchedAckEvent
- AsynchronousServerCallReturnsEvent

⌋()

**[constr_1092] ParameterSwComponentType** ⌈ A ParameterSwComponentType shall never aggregate a SwcInternalBehavior and also owns exclusively PPortPrototype s of type ParameterInterface .

⌋()

**[constr_1093] Definition of textual strings** ⌈ An ApplicationPrimitiveDataType of category STRING shall have a swTextProps which determines the arraySizeSemantics and swMaxTextSize .

⌋()

**[constr_1095] Values of nDataSets vs. reliability** ⌈ If the value of nDataSets is greater than 0 the value of reliability shall not be set to errorCorrection .

⌋()

**[constr_1096] SwcModeSwitchEvent and WaitPoint** ⌈ A RunnableEntity that has a WaitPoint shall not be referenced by a SwcModeSwitchEvent .

⌋()

**[constr_1097] RunnableEntity that has a WaitPoint** ⌈ A RunnableEntity that has a WaitPoint shall not be referenced by a RTEEvent that has a reference in the role disabledMode .

⌋()

**[constr_1098] Mode switch and mode disabling** ⌈ A SwcModeSwitchEvent shall not simultaneously reference to the same ModeDeclaration in both the roles mode and disabledMode .

⌋()

**[constr_1100] Unconnected RPortPrototype typed by a DataInterface** ⌈ For any element in an unconnected RPortPrototype typed by a DataInterface there shall be a requiredComSpec that defines an initValue .

⌋()

**[constr_1101] Mode-related communication** ⌈ An RPortPrototype typed by ModeSwitchInterface shall not be referenced by more than one SwConnector .

⌋()

**[constr_1102] ApplicationError in the scope of one SwComponentType** ⌈ If a SwComponentType has PortPrototype s typed by different ClientServerInterface s with equal shortName and ApplicationError s defined then the following condition applies: ApplicationError s with the same shortName shall have **identical values** of errorCode s.

⌋()

**[constr_1103] NonqueuedReceiverComSpec and enableUpdate** ⌈ A NonqueuedReceiverComSpec that has attribute enableUpdate set to true may not reference a dataElement that in turn is referenced by a VariableAccess in the role dataReadAccess .

⌋()

**[constr_1104] Trigger sink and trigger source** ⌈ An `RPortPrototype` typed by a `TriggerInterface` shall not be referenced by more than one `SwConnector` s that are in turn referencing `PPortPrototype` s typed by `TriggerInterface` s that contain `Trigger` s with the same `shortName` .

⌋()

**[constr_1105] Value of `arraySize`** ⌈ The value of the attribute `arraySize` of an `ImplementationDataTypeElement` owned by an `ImplementationDataType` or `ImplementationDataTypeElement` of `category ARRAY` shall be greater than 0 unless attribute `ImplementationDataTypeElement . arraySizeHandling` exists and is set to the value `inheritedFromArrayElementTypeSize` .

⌋()

**[constr_1106] Structure shall have at least one element** ⌈ An `ImplementationDataType` or `ImplementationDataTypeElement` of `category STRUCTURE` shall own at least one `ImplementationDataTypeElement` .

⌋()

**[constr_1107] Union shall have at least one element** ⌈ An `ImplementationDataType` or `ImplementationDataTypeElement` of `category UNION` shall own at least one `ImplementationDataTypeElement` .

⌋()

**[constr_1108] Value of `ApplicationError . errorCode`** ⌈ The value of `ApplicationError . errorCode` shall not exceed the closed interval 1 .. 63. The following exception applies: **only** in case `possibleError` is supposed to represent `E_OK` the value 0 shall be be allowed.

⌋()

**[constr_1109] Mapping of `SwComponentPrototype` s typed by a `SensorActuatorSwComponentType`** ⌈ A `SwComponentPrototype` typed by a `SensorActuatorSwComponentType` needs to be mapped and run on exactly that ECU that contains the `HwElement` corresponding to the `HwType` that its `SensorActuatorSwComponentType` refers to in case it accesses the hardware via the I/O hardware abstraction layer.

⌋()

**[constr_1110] Value of `category` in `EndToEndDescription`** ⌈ The attribute `category` of `EndToEndDescription` can have the following values:

- NONE
- PROFILE_01
- PROFILE_02

⌋()

**[constr_1111] Constraints of `dataId` in PROFILE_01** ⌈ In PROFILE_01, there shall be only one element in the set and the applicable range of values is [0 .. 65535].

⌋()

**[constr_1112] Constraints of `dataIdMode` in PROFILE_01** ⌈ In PROFILE_01, the applicable range of values for `dataIdMode` is [0 .. 3].

⌋()

**[constr_1113] Existence of attributes in PROFILE_01** ⌈ In PROFILE_01, the following attributes shall exist:

- `dataLength`
- `dataId`

⌋()

**[constr_1114] Constraints of `crcOffset` in PROFILE_01** ⌈ In PROFILE_01, the applicable range of values for `crcOffset` is [0 .. 65535]. For the value of this attribute the constraint *value mod 4 = 0* applies.

⌋()

**[constr_1115] Constraints of `counterOffset` in PROFILE_01** ⌈ In PROFILE_01, the applicable range of values for `counterOffset` is [0 .. 65535]. For the value of this attribute the constraint *value mod 4 = 0* applies.

⌋()

**[constr_1116] Constraints of `dataLength` in PROFILE_01** ⌈ In PROFILE_01, the applicable range of values for `dataLength` is [0 .. 240]. For the value of this attribute the constraint *value mod 8 = 0* applies.

⌋()

**[constr_1117] Constraints of `maxDeltaCounterInit` in PROFILE_01** ⌈ In PROFILE_01, the applicable range of values for `EndToEndDescription.maxDeltaCounterInit` and `ReceiverComSpec.maxDeltaCounterInit` is [0 .. 14].

⌋()

**[constr_1118] Existence of attributes in PROFILE_02** ⌈ In PROFILE_02, only the following attributes shall exist:

- `dataLength`
- `dataId`

⌋()

**[constr_1119] Constraints of `dataLength` in PROFILE_02** ⌈ In PROFILE_02, the applicable range of values for `dataLength` is [0 .. 65535]. For the value of this attribute the constraint *value mod 8 = 0* applies.

⌋()

**[constr_1120] Constraints of `dataId` in PROFILE_02** ⌈ In PROFILE_02, there shall be exactly ordered 16 elements in the set and the applicable range of values is [0 .. 255].

⌋()

**[constr_1121] Constraints of `maxDeltaCounterInit` in PROFILE_02** ⌈ In PROFILE_02, the applicable range of values for `EndToEndDescription.maxDeltaCounterInit` and `ReceiverComSpec.maxDeltaCounterInit` is [0 .. 15].

⌋()

**[constr_1126] Compatibility of `DataConstr`s** ⌈ The `DataConstr` (e.g. the limits) defined by the type of the providing data element shall be within the constraints defined by the type of the requiring data element.

For client-server communication, the following rules apply:

- For `argument`s with attribute `direction` set to the value `in`, the client shall take the role of the *provider* and the server shall take the role of the *requiring side*.
- For `argument`s with attribute `direction` set to the value `inout` the `DataConstr` shall be equal on both sides.
- For `argument`s with attribute `direction` set to the value `out`, the server shall take the role of the *provider* and the client shall take the role of the *requiring side*.

⌋()

**[constr_1128] Queue length of `ClientServerOperation`s associated with the same `RunnableEntity`** ⌈ If two or more `OperationInvokedEvent`s reference a single `RunnableEntity` the value of the `ServerComSpec` attribute `queueLength` shall be **identical** for all `ServerComSpec`s owned by `PPortPrototype`s of the enclosing `SwComponentType` that reference one of the `ClientServerOperation`s that are also referenced by the `OperationInvokedEvent`s.

⌋()

**[constr_1129] `swImplPolicy` and `NonqueuedReceiverComSpec`** ⌈ The attribute `swImplPolicy` of a `dataElement` referenced by a `NonqueuedReceiverComSpec` **shall not** be set to the value `queued`.

⌋()

**[constr_1130] `swImplPolicy` and `QueuedReceiverComSpec`** ⌈ The attribute `swImplPolicy` of a `dataElement` referenced by a `QueuedReceiverComSpec` **shall** be set to the value `queued` .

⌋()

**[constr_1131] `swImplPolicy` and `NonqueuedSenderComSpec`** ⌈ The attribute `swImplPolicy` of a `dataElement` referenced by a `NonqueuedSenderComSpec` **shall not** be set to the value `queued` .

⌋()

**[constr_1132] `swImplPolicy` and `QueuedSenderComSpec`** ⌈ The attribute `swImplPolicy` of a `dataElement` referenced by a `QueuedSenderComSpec` **shall** be set to the value `queued` .

⌋()

**[constr_1134] Allowed structure of `TEXTTABLE`** ⌈ `physConstrs` is not allowed. `compuInternalToPhys` shall exist with `compuScale` s consisting of `upperLimit` and `lowerLimit` .

⌋()

**[constr_1135] Limit of `vt` in `BITFIELD_TEXTTABLE`** ⌈ The separator is "|" and is forbidden in `vt` therefore.

⌋()

**[constr_1137] Applicability of `ParameterInterface`** ⌈ A `PPortPrototype` typed by a `ParameterInterface` can **only** be owned by a `ParameterSwComponentType` .

⌋()

**[constr_1138] `assignedPort` and `DiagEventDebounceMonitorInternal`** ⌈ The existence of an `assignedPort` in combination with a `DiagEventDebounceAlgorithm` shall only be respected for the concrete subclass `DiagEventDebounceMonitorInternal` .

⌋()

**[constr_1139] `assignedPort` of `DiagEventDebounceMonitorInternal` shall refer to an `RPortPrototype`** ⌈ Concerning the debouncing, the software-component acts as a client and thus the `assignedPort` defined with respect to a `DiagEventDebounceMonitorInternal` may only refer to an `RPortPrototype` . The standardized value of the `role` identifier of the `assignedPort` shall be `DiagFaultDetectionCounterPort` .

⌋()

**[constr_1140] Combination of `invalidValue` with the attribute `handleInvalid`** ⌈ The combination of setting the attribute `handleInvalid` of the meta-class `Inval-`

idationPolicy owned by SenderReceiverInterface to value replace **and** of setting the value of the attribute initValue owned by a corresponding NonqueuedReceiverComSpec effectively to the value of the invalidValue (owned by a corresponding SwDataDefProps ) is not supported.

⌋()

**[constr_1141] Applicability of the scope attribute** ⌈ The attribute scope of meta-class VariableAccess shall **only** be applied with respect to the aggregation of VariableAccess in the following roles:

- dataReadAccess
- dataWriteAccess
- dataSendPoint
- dataReceivePointByValue
- dataReceivePointByArgument

⌋()

**[constr_1142] category of CompuMethod shall not be extended** ⌈ In contrast to the general rule that category can be extended by user-specific values it is **not allowed** to extend the meaning of the attribute category of meta-class CompuMethod

⌋()

**[constr_1143] category of AutosarDataType shall not be extended** ⌈ In contrast to the general rule that category can be extended by user-specific values it is **not allowed** to extend the meaning of the attribute category of meta-class AutosarDataType

⌋()

**[constr_1144] SensorActuatorSwComponentType , EcuAbstractionSwComponentType , and ComplexDeviceDriverSwComponentType may only reference a HwType** ⌈ The attribute sensorActuator of SensorActuatorSwComponentType , the attribute hardwareElement of EcuAbstractionSwComponentType , and the attribute hardwareElement of ComplexDeviceDriverSwComponentType may **only** reference a HwType . References to other subclasses of HwDescriptionEntity are not allowed.

⌋()

**[constr_1146] Applicability of a symbol for a CompuScale in C code** ⌈ The symbol attribute shall only be provided for CompuScale s where the category of the enclosing CompuMethod is one of the following:

- SCALE_LINEAR_AND_TEXTTABLE
- SCALE_RATIONAL_AND_TEXTTABLE

- TEXTTABLE
- BITFIELD_TEXTTABLE

⌋()

**[constr_1147] Standardized values for the attribute category of meta-class PortGroup** ⌈ The following values of the attribute category of meta-class PortGroup are reserved by the AUTOSAR standard:

- MODE_MANAGEMENT : This represents the usage of the PortGroup for the purpose of mode management
- PARTIAL_NETWORKING : This represents the usage of the PortGroup for the purpose of partial networking

⌋()

**[constr_1148] PortInterface s of PortPrototype s used to connect to NvBlockSwComponentType s** ⌈ PortInterface s of PortPrototype s used to connect to NvBlockSwComponentType s as well as the PortInterface s used in the context of NvBlockSwComponentType s shall **always** set the value of the attribute isService to false .

⌋()

**[constr_1149] PortPrototype s used for NV data management** ⌈ A PortPrototype typed by a ClientServerInterface used for NV data management, i.e. the interaction of ApplicationSwComponentType s with NvBlockSwComponentType s, shall be typed by ClientServerInterface s that are compatible to the particular ClientServerInterface s derived from MOD_GeneralBlueprints [9] . constr_1148 applies.

⌋()

**[constr_1150] Usage of valueType for PortDefinedArgumentValue** ⌈ The valueType (typically this boils down to integer values used to specify an “id”) associated with PortDefinedArgumentValue shall be of category VALUE or TYPE_REFERENCE . The latter case is only supported if the value of category of the target data type is set to VALUE .

⌋()

**[constr_1151] Applicability of PortInterfaceMapping** ⌈ A PortInterfaceMapping is only applicable and valid for a SwConnector if the two PortPrototype s which are referenced by the SwConnector are typed by the same two PortInterface s which are mapped by the PortInterfaceMapping .

⌋()

**[constr_1152] category of ApplicationArrayElement and AutosarDataType referenced in the role type shall be kept in sync** ⌈ The value of category of an ApplicationArrayElement shall always be identical to the value of

category of the AutosarDataType referenced by the ApplicationArrayElement .

⌋()

**[constr_1153] Applicability of compatibility requirements for CompuScale s** ⌈ Compatibility requirements for CompuScale s shall only apply for CompuScale s where the category of the enclosing CompuMethod is one of the following:

- SCALE_LINEAR_AND_TEXTTABLE
- SCALE_RATIONAL_AND_TEXTTABLE
- TEXTTABLE
- TAB_NOINTP
- BITFIELD_TEXTTABLE
- LINEAR
- RAT_FUNC
- IDENTICAL

⌋()

**[constr_1154] Compatibility of CompuScale s for sender-receiver communication and similar use cases** ⌈ For sender-receiver communication and similar use cases, it is required that the set of CompuScale s defined in the CompuMethod of the provider of the communication (i.e. on the side of the PPortPrototype ) shall be a subset of the set of CompuScale s defined in the CompuMethod on the required side (i.e. on the side of the RPortPrototype ).

⌋()

**[constr_1155] Compatibility of CompuScale s for client-server communication** ⌈ For client-server communication, the following rules apply:

For argument s of direction IN the CompuScale s defined in the CompuMethod of the client (i.e. on the side of the RPortPrototype ) shall be a subset of the set of CompuScale s defined in the CompuMethod supported at the server (i.e. on the side of the PPortPrototype ).

For argument s of the direction OUT the set of CompuScale s defined in the CompuMethod of the server (i.e. on the side of the PPortPrototype ) shall be a subset of the set of CompuScale s defined in the CompuMethod supported at the client (i.e. on the side of the RPortPrototype ).

For argument s of direction INOUT the set of CompuScale s defined in the CompuMethod of server and client shall be identical.

⌋()

**[constr_1156] Relevance of "names" of `CompuScale` s** ⌈ `CompuScale` s which contribute to tabular conversion by having a `compuConst` are compatible **if and only if** the "names" of the `compuScale` s, (namely `shortLabel` , `compuConst` and `symbol` ) are equal. If the scale has no `compuConst` , "names" of `CompuScale` s are not relevant for compatibility.

⌋()

**[constr_1157] Applicability of constraints of `CompuScale` s** ⌈ The constraints constr_1154 , constr_1155 , and constr_1156 shall **only** apply in the absence of a `TextTableMapping` which shall take precedence regarding the compatibility if it exists.

⌋()

**[constr_1158] Applicable `category` s for attribute `ImplementationDataType.swDataDefProps.compuMethod`** ⌈ The definition of the reference `ImplementationDataType.swDataDefProps.compuMethod` is restricted to a `CompuMethod` of either `category BITFIELD_TEXTTABLE` or `category TEXTTABLE` (these might be seen as implementation specific in certain cases).

⌋()

**[constr_1159] Consistency of `VariableAndParameterInterfaceMapping` with respect to the referenced `DataInterface` s** ⌈ Within one `VariableAndParameterInterfaceMapping` all `firstDataPrototype` s shall belong to one and only one `DataInterface` and all `secondDataPrototype` s shall belong to one other and only one other `DataInterface` .

⌋()

**[constr_1160] Size of `Compound Primitive Data Type` is variant** ⌈ For `Compound Primitive Data Type` s (see TPS_SWCT_01179 ) where the size is subject to variation the size of the specified `initValue` s shall match the range of the involved `SwSystemconst` .

⌋()

**[constr_1161] Applicability of the `index` attribute of `Ref`** ⌈ The `index` attribute of `Ref` is limited to a given set if use cases as there are:

- `McDataInstance.instanceInMemory`
- `AutosarVariableRef`
- `AutosarParameterRef`
- `FlatInstanceDescriptor` / `AnyInstanceRef`

⌋()

**[constr_1162] Compatibility of `SwRecordLayout` s** ⌈ Two `SwRecordLayout` definitions are compatible if and only if all attributes **except**

- shortName
- desc
- introduction
- longName
- adminData
- annotation

are **identical** .

⌋()

**[constr_1163] Compatibility of `CompuMethod` s** ⌈ Two CompuMethod definitions are compatible if and only if all attributes **except**

- shortName
- desc
- introduction
- longName
- adminData
- annotation
- displayFormat

are **identical and** the compuScale s and unit s are compatible.

⌋()

**[constr_1164] Number of `argument` s owned by a `RunnableEntity`** ⌈ If a given RunnableEntity owns RunnableEntityArgument s in the role argument , then the number of these RunnableEntityArgument s shall be identical to the number of applicable portArgValue s of the PortAPIOption that references the PortPrototype that in turn is referenced by the OperationInvokedEvent that references the RunnableEntity **plus** the number of ArgumentDataPrototype s aggregated in the role argument by the ClientServerOperation referenced by said OperationInvokedEvent .

⌋()

**[constr_1165] Applicability of `RunnableEntityArgument`** ⌈ The existence of a RunnableEntityArgument is limited to RunnableEntity s triggered by a ClientServerOperation .

⌋()

**[constr_1166] Restrictions of `ModeRequestTypeMap`** ⌈ For every ModeDeclarationGroup referenced by a ModeDeclarationGroupPrototype used in a Port-

Prototype typed by a ModeSwitchInterface a ModeRequestTypeMap shall exist that points to the ModeDeclarationGroup and also to an eligible ImplementationDataType .

The ModeRequestTypeMap shall be aggregated by a DataTypeMappingSet which is referenced from the SwcInternalBehavior that is owned by the ApplicationSwComponentType that also owns the PortPrototype .

⌋()

**[constr_1167] ImplementationDataType s used as ModeRequestTypeMap . implementationDataType** ⌈ The ImplementationDataType referenced by a ModeRequestTypeMap shall **either** be of category VALUE **or** of category TYPE_REFERENCE that in turn references an ImplementationDataType of category VALUE .

The baseType referenced by the ImplementationDataType shall have set the value of the attribute BaseTypeDirectDefinition . baseTypeEncoding to NONE .

⌋()

**[constr_1168] Compatibility of ImplementationDataType s used in the ModeRequestTypeMap** ⌈ Both ImplementationDataType s shall fulfill constr_1167 .

In addition to that, the possible numbers used for representing ModeDeclaration s on the side of the mode manager shall match the supported range of the ImplementationDataType used for representing ModeDeclaration s on the side of the mode user (see constr_1075 ).

⌋()

**[constr_1169] Allowed values for Trigger . swImplPolicy** ⌈ The **only** allowed values for the attribute Trigger . swImplPolicy are either STANDARD (in which case the Trigger processing does not use a queue) or QUEUED (in which case the processing of Trigger s positively uses a queue).

⌋()

**[constr_1170] Interpretation of attribute maxDeltaCounterInit owned by EndToEndDescription** ⌈ If EndToEndProtection . endToEndProtectionVariablePrototype . receiver is identical to the RPortPrototype . requiredComSpec . dataElement **and** RPortPrototype . requiredComSpec . maxDeltaCounterInit is defined **then** the value of RPortPrototype . requiredComSpec . maxDeltaCounterInit **shall be preferred** over the value of EndToEndProtection . endToEndProfile . maxDeltaCounterInit .

If the value of category of EndToEndDescription is set to PROFILE_01 **and either** the described correspondence rule concerning the referenced VariableDataPrototype is not fulfilled **or** RPortPrototype . requiredComSpec . maxDelta-

CounterInit is not defined **then** EndToEndProtection . endToEndProfile . maxDeltaCounterInit **shall exist** .

⌋()

**[constr_1171] Interpretation of attribute maxDeltaCounterInit of EndToEndDescription** ⌈ If EndToEndProtection . endToEndProtectionVariablePrototype . receiver is identical to the RPortPrototype . requiredComSpec . dataElement **and** RPortPrototype . requiredComSpec . maxDeltaCounterInit is defined **then** the value of RPortPrototype . requiredComSpec . maxDeltaCounterInit **shall be preferred** over the value of EndToEndProtection . endToEndProfile . maxDeltaCounterInit .

If the value of category of EndToEndDescription is set to PROFILE_02 **and either** the described correspondence rule concerning the referenced VariableDataPrototype is not fulfilled **or** RPortPrototype . requiredComSpec . maxDeltaCounterInit is not defined **then** EndToEndProtection . endToEndProfile . maxDeltaCounterInit **shall exist** .

⌋()

**[constr_1172] Allowed values of SwCalibrationAccessEnum for ModeDeclarationGroupPrototype** ⌈ The only allowed values of swCalibrationAccess aggregated by ModeDeclarationGroupPrototype are notAccessible and readOnly .

⌋()

**[constr_1173] Applicability of AutosarParameterRef referencing a VariableDataPrototype** ⌈ A reference from AutosarParameterRef to VariableDataPrototype is **only** applicable if the AutosarParameterRef is used in the context of SwAxisGrouped .

⌋()

**[constr_1174] PortInterface s used in the context of CompositionSwComponentType s cannot refer to AUTOSAR services** ⌈ CompositionSwComponentType s shall not own PortPrototype s typed by PortInterface s where the attribute isService is set to true .

⌋()

**[constr_1175] Depending on its category , CompuMethod shall refer to a unit** ⌈ As a CompuMethod specifies the conversion between the physical world and the numerical values they shall refer to a unit unless the CompuMethod 's category is one of TEXTTABLE , BITFIELD_TEXTTABLE , or IDENTICAL .

⌋()

**[constr_1176] Compatibility of CompuScale s of category LINEAR and RAT_FUNC** ⌈ CompuScale s of category LINEAR and RAT_FUNC are considered compatible if they yield the same conversion.

⌋()

**[constr_1177] Allowed `targetCategory` for `SwPointerTargetProps`** ⌈ The value of `targetCategory` for `SwPointerTargetProps` can only be one of `TYPE_REFERENCE` or `FUNCTION_REFERENCE` . The only exception from this rule applies if the `swDataDefProps` owned by the `SwPointerTargetProps` refers to a `SwBaseType` with native type declaration `void` , in this case the value `VALUE` is also permitted.

⌋()

**[constr_1178] Existence of attributes of `SwDataDefProps` in the context of `ImplementationDataType`** ⌈ For the sake of removing possible sources of ambiguity, `SwDataDefProps` used in the context of `ImplementationDataType` can **only have one of**

- `baseType`
- `swPointerTargetProps`
- `implementationDataType`

⌋()

**[constr_1181] Numerical values used in `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue`** ⌈ The numerical values used to define the `value` attributes and the `onTransitionValue` attribute of a `ModeDeclarationGroup` shall not overlap.

⌋()

**[constr_1182] Allowed values for `InternalTriggeringPoint.swImplPolicy`** ⌈ The **only** allowed values for the attribute `swImplPolicy` of meta-class `InternalTriggeringPoint` are either `STANDARD` (in which case the processing of the internal triggering does not use a queue) or `QUEUED` (in which case the processing of internal triggering positively uses a queue).

⌋()

**[constr_1183] `EndToEndProtectionVariablePrototype`s aggregated by `EndToEndProtection`** ⌈ All `EndToEndProtectionVariablePrototype` s aggregated by the same `EndToEndProtection` shall refer to the identical `sender` .

⌋()

**[constr_1184] Consistency of `rootDataPrototype` and `base` in the context of `ApplicationCompositeElementInPortInterfaceInstanceRef`** ⌈ The `rootDataPrototype` referenced by `ApplicationCompositeElementInPortInterfaceInstanceRef` shall be owned by the applicable subclass of `DataInterface` referenced in the role `base` .

This implies that the rootDataPrototype shall be a ParameterDataPrototype if the base is a ParameterInterface . Otherwise the rootDataPrototype shall be a VariableDataPrototype .

⌋()

**[constr_1185] Consistency of data types in the context of ApplicationCompositeElementInPortInterfaceInstanceRef** ⌈ The definition of attributes contextDataPrototype and targetDataPrototype shall (via the type-prototype pattern) be enclosed in the context of the definition of the data type used to type rootDataPrototype .

⌋()

**[constr_1186] Consistency of data types in the context of ArVariableInImplementationDataInstanceRef** ⌈ The definition of attributes contextDataPrototype and targetDataPrototype shall be enclosed in the context of the definition of the data type used to type rootVariableDataPrototype .

⌋()

**[constr_1187] Compatibility of VariableDataPrototype s or ParameterDataPrototype s typed by composite data types** ⌈ DataPrototype s of ApplicationCompositeDataType s or ImplementationDataType s of category STRUCTURE or ARRAY are compatible if one of the following conditions evaluates to true:

1. The underlying ApplicationCompositeDataType s or ImplementationDataType s of category STRUCTURE or ARRAY are identical
2. The underlying ApplicationCompositeDataType s or ImplementationDataType s of category STRUCTURE or ARRAY fulfill the following condition:
   - They consist of the same number of elements **and**
   - They are composed of compatible AutosarDataType s (either ApplicationCompositeDataType s or ImplementationDataType s of category STRUCTURE or ARRAY **OR** ApplicationPrimitiveDataType s or ImplementationDataType s of category VALUE , BOOLEAN , or STRING ) in *the same order* **and**
   - All attributes match exactly, with the exception of the shortName of the M1 AutosarDataType .
3. In the context of a DataPrototypeMapping , for each ApplicationCompositeElementDataPrototype of the required DataPrototype a SubElementMapping exists such that a ApplicationCompositeDataTypeSubElementRef in the role firstElement or secondElement exists that references the required ApplicationCompositeElementDataPrototype **and** a corresponding ApplicationCompositeDataTypeSubElementRef exists in the **other** role (i.e. secondElement or firstElement ) that in turn refer-

ences an ApplicationCompositeElementDataPrototype of the provided ApplicationCompositeDataType .

4. If and only if the DataPrototype is **not** typed by an ApplicationDataType but by an ImplementationDataType : in the context of a DataPrototypeMapping , for each ImplementationDataTypeElement of the required DataPrototype a SubElementMapping exists such that a ImplementationDataTypeSubElementRef in the role firstElement or secondElement exists that references the required ImplementationDataTypeElement **and** a corresponding ImplementationDataTypeSubElementRef exists in the **other** role (i.e. secondElement or firstElement ) that in turn references an ImplementationDataTypeElement of the provided ImplementationDataType .

⌋()

**[constr_1188] Existence of ReceiverComSpec.replaceWith** ⌈ The aggregation of VariableAccess in the role ReceiverComSpec.replaceWith shall exist **if and only if at least one of the following conditions is fulfilled** :

- Attribute ReceiverComSpec.handleOutOfRange is set to the value externalReplacement .
- Attribute SenderReceiverInterface.invalidationPolicy.handleInvalid is set to the value externalReplacement .

⌋()

**[constr_1190] Only one mapping for composite to primitive use case** ⌈ In the case described by TPS_SWCT_01195 only one subElementMapping shall exist at the enclosing DataPrototypeMapping .

⌋()

**[constr_1191] Value of Limit shall yield a numerical value** ⌈ After all variability is bound, the content obtained from a limit shall yield a numerical value.

⌋()

**[constr_1192] Compatibility of " IDENTICAL " to " RAT_FUNC " or " LINEAR "** ⌈ Similar to constr_1176 , a CompuScale where the category of the enclosing CompuMethod is set to IDENTICAL is considered compatible to a CompuScale where the category of the enclosing CompuMethod is set to RAT_FUNC or LINEAR if the following rule applies:

$$int = \frac{N_0+N_1*phys+N_i*phys^i}{D_0+D_1*phys+D_i*phys^i} = phys$$

⌋()

**[constr_1193] ModeDeclaration shall be referenced by at least one ModeTransition in the role enteredMode** ⌈ For each ModeDeclaration at least one ModeTransition shall reference the ModeDeclaration in the role enteredMode .

This constraint shall apply **only** if there is at least one ModeTransition defined in the context of the enclosing ModeDeclarationGroup and it shall **not** apply to the initialMode .

⌋()

**[constr_1194] Identical ModeTransition s** ⌈ Two ModeDeclarationGroup s contain identical modeTransition s if and only if

1. For each ModeTransition defined in the context of the mode provider one ModeTransition with the same shortName is defined in the context of the mode user.
2. Each pair of ModeTransition s in both ModeDeclarationGroup s identified by their respective shortName have identical targets (in terms of the shortName of the referenced ModeDeclaration ) of the references enteredMode and exitedMode .

⌋()

**[constr_1195] SwcModeSwitchEvent and the definition of ModeTransition** ⌈ For each pair of ModeDeclaration s referenced by a SwcModeSwitchEvent with attribute activation set to onTransition a ModeTransition shall be defined in the corresponding direction (i.e. from exitedMode to enteredMode ). This constraint shall only apply if the respective ModeDeclarationGroup defines at least one modeTransition .

⌋()

**[constr_1196] Existence of networkRepresentation vs. compositeNetworkRepresentation** ⌈ If a ReceiverComSpec or SenderComSpec aggregates networkRepresentation it shall **not** aggregate compositeNetworkRepresentation at the same time (and vice versa).

⌋()

**[constr_1197] Existence of compositeNetworkRepresentation shall be comprehensive** ⌈ If at least one compositeNetworkRepresentation exists then for each leaf ApplicationCompositeElementDataPrototype of the affected ApplicationCompositeDataType exactly one compositeNetworkRepresentation shall be defined.

⌋()

**[constr_1200] Queued communication is not applicable for dataElement s owned by PRPortPrototype** ⌈ The swImplPolicy shall not be set to queued for any dataElement owned by a PRPortPrototype .

⌋()

**[constr_1202] Supported connections by AssemblySwConnector for PortPrototype s typed by a SenderReceiverInterface or NvDataInterface** ⌈ For

the modeling of AssemblySwConnector s between PortPrototype s typed by a SenderReceiverInterface or NvDataInterface , **only** the connections documented in Table table_3a_supportedAssSRNVConnections are supported by AUTOSAR.

⌋()

**[constr_1203] Supported connections by DelegationSwConnector for PortPrototype s typed by a SenderReceiverInterface or NvDataInterface** ⌈ For the modeling of DelegationSwConnector s between PortPrototype s typed by a SenderReceiverInterface or NvDataInterface , **only** the connections documented in Table table_3a_supportedDelSRNVConnections are supported by AUTOSAR.

⌋()

**[constr_1204] Supported connections by AssemblySwConnector for PortPrototype s typed by a ClientServerInterface , ModeSwitchInterface , or TriggerInterface** ⌈ For the modeling of AssemblySwConnector s between PortPrototype s typed by a ClientServerInterface , ModeSwitchInterface , or TriggerInterface , **only** the connections documented in Table table_3a_supportedAssCSMTConnections are supported by AUTOSAR.

⌋()

**[constr_1205] Supported connections by DelegationSwConnector for PortPrototype s typed by a ClientServerInterface , ModeSwitchInterface , or TriggerInterface** ⌈ For the modeling of DelegationSwConnector s between PortPrototype s typed by a ClientServerInterface , ModeSwitchInterface , or TriggerInterface , **only** the connections documented in Table table_3a_supportedDelCSMTConnections are supported by AUTOSAR.

⌋()

**[constr_1209] Mapping of ModeDeclaration s of mode user to ModeDeclaration of mode manager** ⌈ A configuration that maps **several** ModeDeclaration s representing modes of a mode user to **one** ModeDeclaration representing a mode of a mode manager shall be rejected.

⌋()

**[constr_1210] Mapping of ModeDeclaration s of mode user to all ModeDeclaration s of mode manager** ⌈ If a ModeDeclarationMapping exists that references a ModeDeclaration representing a mode of the mode manager then ModeDeclarationMapping s shall exist that map all modes of the mode manager to modes of the mode user.

⌋()

**[constr_1211] Constraints of maxNoNewOrRepeatedData in PROFILE_01** ⌈ In PROFILE_01, the applicable range of values for EndToEndDescription .

maxNoNewOrRepeatedData and ReceiverComSpec . maxNoNewOrRepeated-Data is [0 .. 14].

⌋()

**[constr_1212] Constraints of syncCounterInit in PROFILE_01** ⌈ In PROFILE_01, the applicable range of values for EndToEndDescription . syncCounterInit and ReceiverComSpec . syncCounterInit is [0 .. 14].

⌋()

**[constr_1213] Constraints of maxNoNewOrRepeatedData in PROFILE_02** ⌈ In PROFILE_02, the applicable range of values for EndToEndDescription . maxNoNewOrRepeatedData and ReceiverComSpec . maxNoNewOrRepeated-Data is [0 .. 15].

⌋()

**[constr_1214] Constraints of syncCounterInit in PROFILE_02** ⌈ In PROFILE_02, the applicable range of values for EndToEndDescription . syncCounterInit and ReceiverComSpec . syncCounterInit is [0 .. 15].

⌋()

**[constr_1215] Interpretation of attribute maxNoNewOrRepeatedData owned by EndToEndDescription in PROFILE_01** ⌈ If EndToEndProtection . endToEnd-ProtectionVariablePrototype . receiver is identical to the RPortPrototype . requiredComSpec . dataElement **and** RPortPrototype . requiredComSpec . maxNoNewOrRepeatedData is defined **then** the value of RPortPrototype . requiredComSpec . maxNoNewOrRepeatedData **shall be preferred** over the value of EndToEndProtection . endToEndProfile . maxNoNewOrRepeatedData .

If the value of category of EndToEndDescription is set to PROFILE_01 **and either** the described correspondence rule concerning the referenced Variable-DataPrototype is not fulfilled **or** RPortPrototype . requiredComSpec . maxNoNewOrRepeatedData is not defined **then** EndToEndProtection . end-ToEndProfile . maxNoNewOrRepeatedData **shall exist** .

⌋()

**[constr_1216] Interpretation of attribute syncCounterInit owned by End-ToEndDescription in PROFILE_01** ⌈ If EndToEndProtection . endToEndProtectionVariablePrototype . receiver is identical to the RPortPrototype . requiredComSpec . dataElement **and** RPortPrototype . requiredComSpec . syncCounterInit is defined **then** the value of RPortPrototype . requiredCom-Spec . syncCounterInit **shall be preferred** over the value of EndToEndProtection . endToEndProfile . syncCounterInit .

If the value of category of EndToEndDescription is set to PROFILE_01 **and either** the described correspondence rule concerning the referenced VariableDataPrototype is not fulfilled **or** RPortPrototype . requiredComSpec . sync-

CounterInit is not defined **then** EndToEndProtection . endToEndProfile . syncCounterInit **shall exist** .

⌋()

**[constr_1217] Interpretation of attribute maxNoNewOrRepeatedData owned by EndToEndDescription in PROFILE_02** ⌈ If EndToEndProtection . endToEndProtectionVariablePrototype . receiver is identical to the RPortPrototype . requiredComSpec . dataElement **and** RPortPrototype . requiredComSpec . maxNoNewOrRepeatedData is defined **then** the value of RPortPrototype . requiredComSpec . maxNoNewOrRepeatedData **shall be preferred** over the value of EndToEndProtection . endToEndProfile . maxNoNewOrRepeatedData .

If the value of category of EndToEndDescription is set to PROFILE_02 **and either** the described correspondence rule concerning the referenced VariableDataPrototype is not fulfilled **or** RPortPrototype . requiredComSpec . maxNoNewOrRepeatedData is not defined **then** EndToEndProtection . endToEndProfile . maxNoNewOrRepeatedData **shall exist** .

⌋()

**[constr_1218] Interpretation of attribute syncCounterInit owned by EndToEndDescription in PROFILE_02** ⌈ If EndToEndProtection . endToEndProtectionVariablePrototype . receiver is identical to the RPortPrototype . requiredComSpec . dataElement **and** RPortPrototype . requiredComSpec . syncCounterInit is defined **then** the value of RPortPrototype . requiredComSpec . syncCounterInit **shall be preferred** over the value of EndToEndProtection . endToEndProfile . syncCounterInit .

If the value of category of EndToEndDescription is set to PROFILE_02 **and either** the described correspondence rule concerning the referenced VariableDataPrototype is not fulfilled **or** RPortPrototype . requiredComSpec . syncCounterInit is not defined **then** EndToEndProtection . endToEndProfile . syncCounterInit **shall exist** .

⌋()

**[constr_1219] Invalidation depends on the value of swImplPolicy** ⌈ Invalidation of dataElement s is only supported for dataElement s where the value of swImplPolicy is **not** set to queued .

⌋()

**[constr_1220] Compatibility of SwBaseType** ⌈ Two SwBaseType s are compatible if and only if attributes baseTypeSize respectively byteOrder , memAlignment , baseTypeEncoding , and nativeDeclaration have identical values.

⌋()

**[constr_1221] DataPrototype is typed by an ApplicationPrimitiveDataType** ⌈ If a DataPrototype is typed by an ApplicationPrimitive-

DataType its initValue shall be provided by an ApplicationValueSpecification .

If the underlying ApplicationPrimitiveDataType represents an enumeration, the value provided shall match to one of the applicable text values ( vt , shortLabel , symbol ) defined by the applicable CompuScale s.

⌋()

**[constr_1222] category of an AutosarDataType used to type a DataPrototype is set to STRING** ⌈ If the category of an AutosarDataType used to type a DataPrototype is set to STRING the ApplicationValueSpecification used to initialize the DataPrototype shall be of category STRING .

⌋()

**[constr_1223] DataPrototype is typed by an ApplicationRecordDataType** ⌈ If a DataPrototype is typed by an ApplicationRecordDataType the corresponding initValue shall be provided by a RecordValueSpecification .

⌋()

**[constr_1224] DataPrototype is typed by an ApplicationArrayDataType** ⌈ If a DataPrototype is typed by an ApplicationArrayDataType the corresponding initValue shall be provided by an ArrayValueSpecification or ApplicationRuleBasedValueSpecification .

⌋()

**[constr_1225] DataPrototype is typed by an ImplementationDataType that references a CompuMethod of category TEXTTABLE or BITFIELD_TEXTTABLE** ⌈ If a DataPrototype is typed by an ImplementationDataType that references a CompuMethod of category TEXTTABLE or BITFIELD_TEXTTABLE the applicable ValueSpecification shall be a TextValueSpecification .

In this case the value provided shall match to one of the applicable text values ( vt , shortLabel , symbol ) defined by the applicable CompuScales .

⌋()

**[constr_1226] Applicable range for ExecutableEntityActivationReason . bitPosition** ⌈ The value of attribute ExecutableEntityActivationReason . bitPosition shall be in the range of 0 .. 31.

⌋()

**[constr_1227] Value of attribute ExecutableEntityActivationReason . bitPosition shall be unique** ⌈ The value of attributes ExecutableEntityActivationReason . bitPosition and ExecutableEntityActivationReason . symbol shall be unique in the context of the enclosing RunnableEntity .

⌋()

**[constr_1228] `RTEEvent` that is referenced by a `WaitPoint` in the role `trigger` shall not reference `ExecutableEntityActivationReason`** ⌈ An `RTEEvent` that is referenced by a `WaitPoint` in the role `trigger` shall not reference `ExecutableEntityActivationReason` in the role `activationReasonRepresentation`.

⌋()

**[constr_1229] `category` of `ImplementationDataType` boils down to `VALUE`** ⌈ An `ImplementationDataType` qualifies as an `Integral Primitive Type` if and only if either

- its `category` is `VALUE` or `TYPE_REFERENCE` that eventually boils down to `VALUE` **or**
- its `category` is `ARRAY` **and** it has only one `subElement` **and** one of the following conditions applies:
  - `subElement.category` is set to `VALUE` or `TYPE_REFERENCE` that eventually boils down to `VALUE` **and** the `subElement` refers to a `SwBaseType` where `baseTypeSize` is set to the value 8 **and** the `baseTypeEncoding` is set to `NONE`.
  - `subElement.category` is set to `TYPE_REFERENCE` **and** the `swDataDefProps.implementationDataType` literally represents the `Platform Data Type` named “uint8”.
  - `subElement.category` is set to `TYPE_REFERENCE` **and** the attribute `swDataDefProps.implementationDataType.shortName` is set to “uint8” **and** `swDataDefProps.baseType.baseTypeDefinition.nativeDeclaration` does not exist.

⌋()

**[constr_1230] `ApplicationDataType` that qualifies for `Integral Primitive Type`** ⌈ An `ApplicationDataType` qualifies as an `Integral Primitive Type` if and only if **all** of the following conditions apply:

- `ApplicationDataType.category` is set to `BOOLEAN`, `VALUE`, `STRING`, or `ARRAY`
- in the applicable scope a `DataTypeMap` is available that refers to the given `ApplicationDataType`
- the found `DataTypeMap` refers to an `ImplementationDataType` that fulfills the requirements of constr_1229

⌋()

**[constr_1231] `ConsistencyNeeds` aggregated by `CompositionSwComponentType`** ⌈ If `ConsistencyNeeds` are aggregated by a `CompositionSwComponent-`

Type the associations stereotyped instanceRef may only refer to context and target elements within the context of this CompositionSwComponentType .

⌋()

**[constr_1232] ConsistencyNeeds aggregated by AtomicSwComponentType** ⌈ If ConsistencyNeeds are aggregated by a AtomicSwComponentType the associations stereotyped instanceRef may only refer to context and target elements within the context of this AtomicSwComponentType .

⌋()

**[constr_1233] InstantiationTimingEventProps shall only reference TimingEvent** ⌈ An InstantiationTimingEventProps shall only reference TimingEvent in the role refinedEvent . A reference to other kinds of RTEEvent s is not supported.

⌋()

**[constr_1234] Value of RunnableEntity . symbol** ⌈ The value of a RunnableEntity . symbol owned by an NvBlockSwComponentType that is triggered by an OperationInvokedEvent shall only be taken from the set of API names associated with the NvM .

⌋()

**[constr_1237] Scope of mapped ClientServerOperation s in the context of a ClientServerOperationMapping** ⌈ All ClientServerOperation s referenced by a ClientServerOperationMapping in the role firstOperation shall belong to exactly one ClientServerInterface .

All ClientServerOperation s referenced by a ClientServerOperationMapping in the role secondOperation shall belong to exactly one other ClientServerInterface .

⌋()

**[constr_1238] Scope of mapped ApplicationError s in the context of a ClientServerOperationMapping** ⌈ All ApplicationError s referenced by a ClientServerApplicationErrorMapping in the role firstApplicationError shall belong to exactly one ClientServerInterface .

All ApplicationError s referenced by a ClientServerApplicationErrorMapping in the role secondApplicationError shall belong to exactly one other ClientServerInterface .

⌋()

**[constr_1240] Consistency of ArgumentDataPrototype s within the context of a ClientServerOperationMapping** ⌈ Unless a ClientServerOperationMapping . firstToSecondDataTransformation exists, for each argument owned by a ClientServerOperationMapping . firstOperation and

ClientServerOperationMapping . secondOperation a reference in the role ClientServerOperationMapping . argumentMapping . firstDataPrototype or ClientServerOperationMapping . argumentMapping . secondDataPrototype shall exist originated by one of the ClientServerOperationMapping . argumentMapping s owned by the mentioned ClientServerOperationMapping .

⌋()

**[constr_1241] Compound Primitive Data Type s and invalidValue** ⌈ Compound Primitive Data Type s that have set the value of of category other than STRING shall **not** define invalidValue .

⌋()

**[constr_1242] Restriction of invalidValue for ApplicationPrimitiveDataType of category STRING** ⌈ invalidValue for ApplicationPrimitiveDataType of category STRING ( constr_1241 applies) is restricted to be either a compatible ApplicationValueSpecification or a ConstantReference that in turn points to a compatible ApplicationValueSpecification .

⌋()

**[constr_1243] NumericalOrText shall either define vf or vt** ⌈ Within the context of one NumericalOrText , **either** the attribute vf **or** the attribute vt shall be defined. The existence of both attributes at the same time is not permitted.

⌋()

**[constr_1244] DataPrototype s used in application software shall not be typed by C enums** ⌈ A DataPrototype that is used in an AtomicSwComponentType shall not set swDataDefProps . additionalNativeTypeQualifier to enum .

⌋()

**[constr_1245] Consideration of ModeTransition s for the compatibility of ModeDeclarationGroup s** ⌈ One of the following conditions for the consideration of ModeTransition s for the compatibility of ModeDeclarationGroup s shall apply:

- **Either** the mode provider **or** the mode user define ModeTransition s.
- The ModeTransition s defined in the context of the mode provider are **identical** to the ModeTransition s defined in the context of the mode user **or** a ModeDeclarationMapping mapping is applied.

⌋()

**[constr_1246] Consistency of firstMode and secondMode in the scope of one ModeDeclarationMappingSet** ⌈ Within the scope of one ModeDeclarationMappingSet , all firstMode s shall belong to one and only one ModeDeclarationGroup and all secondMode s shall belong to one and only one **other** ModeDeclarationGroup

⌋()

**[constr_1247] Consistency of `ModeDeclarationMappingSet` with respect to the referenced `firstModeGroup` and `secondModeGroup`** ⌈ If a `ModeDeclarationGroupPrototypeMapping . modeDeclarationMappingSet` exists, the `ModeDeclarationGroup` owning the `modeDeclaration` s referenced in the role `firstMode` shall be the `type` of the `ModeDeclarationGroupPrototypeMapping . firstModeGroup` and the `ModeDeclarationGroup` owning the `modeDeclaration` s referenced in the role `secondMode` shall be the `type` of the `ModeDeclarationGroupPrototypeMapping . secondModeGroup` .

⌋()

**[constr_1248] Compatibility of `PortPrototype` s of different `DataInterface` s in the context of a `PassThroughSwConnector`** ⌈ `PortPrototype` s of different `DataInterface` s are considered compatible if and only if

1. For **at least one** `VariableDataPrototype` or `ParameterDataPrototype` defined in the context of the `DataInterface` of the required outer `PortPrototype` a compatible `VariableDataPrototype` or `ParameterDataPrototype` exists in the `DataInterface` of the provided outer `PortPrototype` .

   The table `tab_3a_Overview_20_of_20_compatibility_20_of_20_ParameterDataP` defines which elements of `PortInterface` are considered compatible depending on the type of `PortInterface` as well as the attribute `swImplPolicy` of the elements of `PortInterface` s.

   **Either** the `shortName` of `VariableDataPrototype` s and `ParameterDataPrototype` s are used to identify the pair **or** a `PortInterfaceMapping` exists that defines which differently named elements of `PortInterface` s correlate with each other.

2. For each such pair, the values of the `PortInterface . isService` attributes are identical.

⌋()

**[constr_1249] Compatibility of `ModeSwitchInterface` s in the context of a `PassThroughSwConnector`** ⌈ `PortPrototype` s of different `ModeSwitchInterface` s are considered compatible if and only if

1. For the `ModeDeclarationGroupPrototype` defined in the context of the `ModeSwitchInterface` of the required outer `PortPrototype` a compatible `ModeDeclarationGroupPrototype` exists in the `ModeSwitchInterface` of the provided outer `PortPrototype` .

   **Either** the `shortName` s of the `ModeDeclarationGroupPrototype` s are used to identify the pair **or** a `ModeInterfaceMapping` exists that maps the corresponding `ModeDeclarationGroupPrototype` s.

2. For each such pair, the values of the `PortInterface . isService` attributes are identical.

⌋()

**[constr_1250] Compatibility of `ClientServerInterface` s in the context of a `PassThroughSwConnector`** ⌈ `PortPrototype` s of different `ClientServerInterface` s are considered compatible if and only if

1. For **at least one** `ClientServerOperation` defined in the context of the `ClientServerInterface` of the provided outer `PortPrototype` a compatible `ClientServerOperation` exists in the `ClientServerInterface` of the required outer `PortPrototype` . **Either** the `shortName` s of the `ClientServerOperation` s are used to identify the pair **or** a `ClientServerInterfaceMapping` exists that maps the corresponding `ClientServerOperation` s.
2. For each such pair, the values of the `PortInterface . isService` attributes are identical.

⌋()

**[constr_1251] Compatibility of `PortPrototype` s of `TriggerInterface` s in the context of a `PassThroughSwConnector`** ⌈ `PortPrototype` s of different `TriggerInterface` s are considered compatible if and only if

1. For **at least one** `Trigger` defined in the context of the `TriggerInterface` of the required outer `PortPrototype` a compatible `Trigger` exists in the `TriggerInterface` of the provided outer `PortPrototype` . **Either** the `shortName` of `Trigger` s are used to identify the pair **or** a `TriggerInterfaceMapping` exists that that refers to one of the `Trigger` s in the role `firstTrigger` and to the other in the role `secondTrigger` .
2. For each such pair, the values of the `PortInterface . isService` attributes are identical.

⌋()

**[constr_1252] Creation of a loop involving a `PassThroughSwConnector` is not allowed** ⌈ A `PassThroughSwConnector` is not allowed if the required outer `PortPrototype` is directly or indirectly connected to the provided outer `PortPrototype` without the placement of a `SwComponentPrototype` typed by an `AtomicSwComponentType` in the chain of `SwConnector` s.

⌋()

**[constr_1253] Supported usage of `VariationPointProxy`** ⌈ The allowed multiplicities for attributes of `VariationPointProxy` depending on the applicable binding time and the value of `VariationPointProxy . category` are documented in Table `tab_3a_SupportedUsageOfVariationPointProxy` .

For clarification, the multiplicities of attributes of meta-class `VariationPointProxy` that are **not** explicitly mentioned in a given row of table `tab_3a_SupportedUsageOfVariationPointProxy` shall be interpreted as [0].

⌋()

**[constr_1254] Definition of a pointer to a pointer** ⌈ AUTOSAR does **not** support the definition of a pointer to a pointer by defining an `ImplementationDataType` of `category` `DATA_REFERENCE` that aggregates `SwDataDefProps` in the role `swDataDefProps` that in turn aggregate `SwPointerTargetProps` in the role `swPointerTargetProps` with attribute `targetCategory` set to `DATA_REFERENCE` that in turn aggregates `SwDataDefProps` in the role `swDataDefProps` that aggregates `SwPointerTargetProps` in the role `swPointerTargetProps` that references an `ImplementationDataType` of `category` e.g. `VALUE` .

⌋()

**[constr_1255] `ApplicationPrimitiveDataType` s of `category` `BOOLEAN` and `STRING`** ⌈ If a `Unit` is referenced from within `SwDataDefProps` and/or `PhysConstrs` owned by an `ApplicationPrimitiveDataType` s of `category` `BOOLEAN` and `STRING` it is required that this `Unit` represents a meaningless unit, i.e. the referenced `physicalDimension` shall not define any exponent value other than 0.

⌋()

**[constr_1256] Acknowledgement feedback in n:1 writer case** ⌈ Within the scope of one `SwcInternalBehavior` , it is **not** allowed that two or more aggregated `RunnableEntity` s own either `dataSendPoint` s or `dataWriteAccess` s that in turn point to the identical `accessedVariable` . `autosarVariable` . `targetDataPrototype` **if** the attribute `transmissionAcknowledge` exists in the context of the `SenderComSpec` owned by the `dataSendPoint` . `accessedVariable` . `autosarVariable` . `portPrototype` (or the respective construct for `dataWriteAccess` ) that also refers to said `dataElement` .

⌋()

**[constr_1257] No `WaitPoint` s allowed** ⌈ A `RunnableEntity` referenced by an `InitEvent` in the role `startOnEvent` shall not aggregate a `WaitPoint` .

⌋()

**[constr_1258] Value of `minimumStartInterval` for `RunnableEntity` s triggered by an `InitEvent`** ⌈ The value of the attribute `ExecutableEntity` . `minimumStartInterval` for a `RunnableEntity` s that is triggered by an `InitEvent` shall always be set to 0.

⌋()

**[constr_1259] Aggregation of `AsynchronousServerCallPoint` and `AsynchronousServerCallResultPoint`** ⌈ A `RunnableEntity` referenced by an `InitEvent` in the role `startOnEvent` may aggregate an `AsynchronousServerCallPoint` but it shall not aggregate an `AsynchronousServerCallResultPoint` .

⌋()

**[constr_1260] No mode disabling for `InitEvent` s** ⌈ An `InitEvent` shall not have a reference to a `ModeDeclaration` in the role `disabledMode` .

⌋()

**[constr_1261] Applicability for `EndToEndDescription . dataIdNibbleOffset`** ⌈ `EndToEndDescription . dataIdNibbleOffset` shall be used **only** if `EndToEndDescription . dataIdMode` is set to the value 3 **and** at the same time `EndToEndDescription . category` is set to PROFILE_01.

⌋()

**[constr_1263] Existence of `ModeErrorBehavior . defaultMode`** ⌈ The optional attribute `ModeErrorBehavior . defaultMode` **shall exist** if the value of the attribute `ModeErrorBehavior . errorReactionPolicy` is set to `defaultMode` .

⌋()

**[constr_1264] Iteration along output axis is only supported for `VALUE` and `VAL_BLK`** ⌈ `swRecordLayoutVIndex` in `SwRecordLayoutV` cannot be 0 for any value of `SwRecordLayoutV . category` other than `VALUE` and `VAL_BLK` .

⌋()

**[constr_1268] `ArgumentDataPrototype . direction` shall be preserved in a `ClientServerOperationMapping`** ⌈ Within the context of a `ClientServerOperationMapping` , the value of the argument `ArgumentDataPrototype . direction` of two mapped `ArgumentDataPrototype` shall be identical.

⌋()

**[constr_1269] Number of `argument` s shall be preserved in a `ClientServerOperationMapping`** ⌈ Within the context of a `ClientServerOperationMapping` , the number of `argument` s of `firstOperation` and `secondOperation` shall be identical.

⌋()

**[constr_1270] `ArgumentDataPrototype` shall be mapped only once in a `ClientServerOperationMapping`** ⌈ Within the context of a `ClientServerOperationMapping` , each `argument` shall only be referenced **once** in the role `firstDataPrototype` or `secondDataPrototype` .

⌋()

**[constr_1271] `RecordValueSpecification . field` s shall be identical to the number of `ApplicationRecordDataType . element` s** ⌈ The initialization of an `DataPrototype` typed by an `ApplicationRecordDataType` by means of a `RecordValueSpecification` shall exactly match the structure of the `ApplicationRecordDataType` .

For this means, it is required that the number of RecordValueSpecification . field s shall be identical to the number of ApplicationRecordDataType . element s.

⌋()

**[constr_1272] RecordValueSpecification . field s shall be identical to the number of subElement s of ImplementationDataType of category STRUCTURE** ⌈ The initialization of an DataPrototype typed by an ImplementationDataType of category STRUCTURE by means of a RecordValueSpecification shall exactly match the structure of the ImplementationDataType of category STRUCTURE .

For this means, it is required that the number of RecordValueSpecification . field s shall be identical to the number of ImplementationDataType . subElement s.

⌋()

**[constr_1273] ArrayValueSpecification . element s shall be identical to the value of ApplicationArrayDataType . element . maxNumberOfElements** ⌈ The initialization of DataPrototype typed by an ApplicationArrayDataType by means of an ArrayValueSpecification shall exactly match the structure of the ApplicationArrayDataType regardless of the setting of the attribute ApplicationArrayDataType . element . arraySizeSemantics .

This means that the number of ArrayValueSpecification . element s shall be identical to the value of ApplicationArrayDataType . element . maxNumberOfElements .

⌋()

**[constr_1274] ArrayValueSpecification . element s shall be identical to the value of ImplementationDataType . subElement . arraySize of category ARRAY** ⌈ The initialization of a DataPrototype typed by an ImplementationDataType of category ARRAY by means of an ArrayValueSpecification shall exactly match the structure of the ImplementationDataType regardless of the setting of the attribute ImplementationDataType . subElement . arraySizeSemantics .

This means that the number of ArrayValueSpecification . element s shall be identical to the value of ImplementationDataType . subElement . arraySize .

⌋()

**[constr_1277] SwDataDefProps . swImplPolicy of a VariableDataPrototype referenced by a VariableAccess aggregated in the role dataReceivePointByValue** ⌈ The SwDataDefProps . swImplPolicy of a VariableDataPrototype referenced by a VariableAccess aggregated in the role dataReceivePointByValue shall not be set to queued .

⌋()

**[constr_1278] `PhysConstrs` references a `Unit`** ⌈ `DataConstr` s are only compatible if the `DataConstr . dataConstrRule . physConstrs . unit` are compatible or neither `DataConstr . dataConstrRule . physConstrs . unit` exist.

⌋()

**[constr_1279] Unmapped elements of `ApplicationCompositeDataType` s or `ImplementationDataType` s and the attribute `swImplPolicy`** ⌈ If the attribute `swImplPolicy` is set to `queued` it is not allowed to have unmapped elements of `ApplicationCompositeDataType` s or `ImplementationDataType` s of `category STRUCTURE` or `ARRAY` on the receiver side.

⌋()

**[constr_1280] Unmapped `dataElement` on the receiver side shall have an init Value** ⌈ If elements of `ApplicationCompositeDataType` s or `ImplementationDataType` s of `category STRUCTURE` or `ARRAY` are not considered in a `SubElementMapping` then the enclosing `dataElement` shall have an `initValue` **if** the `NonqueuedReceiverComSpec` is aggregated by an `AbstractRequiredPortPrototype` .

⌋()

**[constr_1281] `invalidValue` is inside the scope of the `compuMethod`** ⌈ If the value of the `invalidValue` of an `ApplicationPrimitiveDataType` of `category VALUE` is supposed to be **inside** the scope of the applicable `CompuMethod` an `ApplicationValueSpecification` is used to describe the `invalidValue` of the `ApplicationPrimitiveDataType` .

⌋()

**[constr_1282] Restriction concerning the usage of `RuleBasedValueSpecification` or a `ReferenceValueSpecification` for the specification of an `invalidValue`** ⌈ The aggregation of a `RuleBasedValueSpecification` or a `ReferenceValueSpecification` for the definition of a `ApplicationPrimitiveDataType . swDataDefProps . invalidValue` is not supported.

⌋()

**[constr_1283] `invalidValue` is outside the scope of the `compuMethod`** ⌈ If the value of the `invalidValue` of an `ApplicationPrimitiveDataType` of `category VALUE` is supposed to be **outside** the scope of the applicable `CompuMethod` a `NumericalValueSpecification` shall be used to describe the `invalidValue` of the `ApplicationPrimitiveDataType` .

⌋()

**[constr_1284] Limitation of the use of `TextValueSpecification`** ⌈ `TextValueSpecification` shall **only** be used in the context of an `AutosarDataType` that references a `CompuMethod` in the role `ImplementationDataType . swDataDefProps . compuMethod` of `category TEXTTABLE` and `BITFIELD_TEXTTABLE` .

⌋()

**[constr_1285] Applicability of roles vs. `PortPrototype` s** ⌈ The aggregation of `AutosarVariableRef` aggregated by `NvBlockDataMapping` in the roles `writtenNvData` , `writtenReadNvData` , or `readNvData` is subject to limitation depending on the applicable subclass of `PortPrototype` :

- The role `writtenNvData` shall only be used if the corresponding `PortPrototype` is a `RPortPrototype`
- The role `writtenReadNvData` shall only be used if the corresponding `PortPrototype` is a `PRPortPrototype`
- The role `readNvData` shall only be used if the corresponding `PortPrototype` is a `PPortPrototype`

⌋()

**[constr_1286] `serverArgumentImplPolicy` and `ArgumentDataPrototype` typed by primitive data types** ⌈ The value of the attribute `ArgumentDataPrototype . serverArgumentImplPolicy` shall **not** be set to `useVoid` for an `ArgumentDataPrototype` of `direction in` that is typed by an `AutosarDataType` that boils down to a primitive C data type (see TPS_SWCT_01565 ).

⌋()

**[constr_1287] Compatibility of `SenderReceiverInterface` s with respect to `invalidationPolicy`** ⌈ `VariableDataPrototype` s defined in the context of the `SenderReceiverInterface` are only compatible if the `invalidationPolicy` s have the same value.

⌋()

**[constr_1288] Allowed Attributes vs. `category` for `DataPrototype` s typed by `ImplementationDataType` s** ⌈ The allowed values per `category` for `DataPrototype` s typed by `ImplementationDataType` s are documented in table `table_3a_CategoriesImpl4DataProt` .

⌋()

**[constr_1289] Allowed Attributes vs. `category` for `DataPrototype` s typed by `ApplicationDataType` s** ⌈ The allowed values of Attributes per `category` for `DataPrototype` s typed by `ApplicationDataType` s are documented in table `table_3a_CategoriesAppl4DataProt` .

⌋()

**[constr_1290] Limitation on the number of `PPortComSpec` s in the context of one `PPortPrototype`** ⌈ Within the context of one `PPortPrototype` there can only be **one** `PPortComSpec` that references a given `dataElement` or `operation` .

⌋()

**[constr_1291] Limitation on the number of `RPortComSpec` s in the context of one `PPortPrototype`** ⌈ Within the context of one `RPortPrototype` , there can only be **one** `RPortComSpec` that references a given `dataElement` or `operation` .

⌋()

**[constr_1292] Limitation on the number of `RPortComSpec` s/ `PPortComSpec` s in the context of one `PRPortPrototype`** ⌈ Within the context of one `PRPortPrototype` , there can only be **one** `RPortComSpec` and **one** `PPortComSpec` that references a given `dataElement` or `operation` .

⌋()

**[constr_1295] `PortInterface` s and `category DATA_REFERENCE`** ⌈ A `DataPrototype` defined in the context of a `PortInterface` used by an `ApplicationSwComponentType` or `SensorActuatorSwComponentType` that is (after potential indirections via `TYPE_REFERENCE` are resolved) either typed by or mapped to an `ImplementationDataType` of `category DATA_REFERENCE` shall only be used if either the provider or the requester of the information represents a `ServiceSwComponentType` , a `ComplexDeviceDriverSwComponentType` , a `ParameterSwComponentType` , or an `NvBlockSwComponentType` , or the `EcuAbstractionSwComponentType` .

⌋()

**[constr_1296] `DataPrototype` s used as `explicitInterRunnableVariable` or `implicitInterRunnableVariable` and `category DATA_REFERENCE`** ⌈ A `VariableDataPrototype` shall not be aggregated by `SwcInternalBehavior` in either the role `explicitInterRunnableVariable` or `implicitInterRunnableVariable` if the `VariableDataPrototype` (after potential indirections via `TYPE_REFERENCE` are resolved) is either typed by or mapped to an `ImplementationDataType` of `category DATA_REFERENCE` .

⌋()

**[constr_1297] Applicability of `serverArgumentImplPolicy` set to `useArrayBaseType`** ⌈ The value of the attribute `ArgumentDataPrototype` . `serverArgumentImplPolicy` shall only be set to `useArrayBaseType` for an `ArgumentDataPrototype` that is typed by an `AutosarDataType` that is (after all `TYPE_REFERENCE` s are resolved) either an `ImplementationDataType` of `category ARRAY` or an `ApplicationDataType` mapped to (after all `TYPE_REFERENCE` s are resolved) an `ImplementationDataType` of `category ARRAY` .

⌋()

**[constr_1298] Existence of attributes if `category` of a `ModeDeclarationGroup` is set to `EXPLICIT_ORDER`** ⌈ The attributes `ModeDeclarationGroup` . `onTransitionValue` and `ModeDeclaration` . `value` (for each `ModeDeclaration` ) shall be set if the `category` of a `ModeDeclarationGroup` is set to `EXPLICIT_ORDER` .

⌋()

**[constr_1299] Existence of attributes if `category` of a `ModeDeclarationGroup` is set to other than `EXPLICIT_ORDER`** ⌈ The attributes `ModeDeclarationGroup.onTransitionValue` or `ModeDeclaration.value` (for any `ModeDeclaration`) shall **not** be set if the `category` of a `ModeDeclarationGroup` is set to any value **other than** `EXPLICIT_ORDER`.

⌋()

**[constr_1300] Primitive `DataPrototype` on the provider side shall not be mapped to element of a composite data type on the requester side** ⌈ The usage of `DataPrototypeMapping` resp. `SubElementMapping` does not support the following configuration:

- The `AutosarDataPrototype` referenced on the provider/client side is typed by an `ApplicationPrimitiveDataType` of `category VALUE` or `ImplementationDataType` of `category VALUE` or `category TYPE_REFERENCE` that eventually resolves to `category VALUE`.
- The `DataPrototypeMapping` aggregates a `subElementMapping` that refers to a `ImplementationDataTypeElement` or `ApplicationCompositeElementDataPrototype` on the requester/server side.

⌋()

**[constr_1301] Existence of `RoleBasedDataTypeAssignment.role` vs. `RoleBasedDataAssignment.role`** ⌈ The usage of a `RoleBasedDataTypeAssignment` with attribute `role` set to the value `temporaryRamBlock` is only allowed if **no** `RoleBasedDataAssignment` defined with attribute `role` set to value `defaultValue` exists in the owning `SwcServiceDependency`.

⌋()

**[constr_1302] Restriction of data invalidation** ⌈ Data invalidation is only applicable for one of the following cases applicable on the **receiving** side:

1. `VariableDataPrototype` s typed by either an `ApplicationPrimitiveDataType` or an `ImplementationDataType` of `category VALUE` or `TYPE_REFERENCE` that boils down to `category VALUE` that have defined an `invalidValue`.
2. `VariableDataPrototype` s typed by either an `ApplicationCompositeDataType` or an `ImplementationDataType` of `category STRUCTURE`, or `ARRAY` or of `category TYPE_REFERENCE` that boils down to `category STRUCTURE`, or `ARRAY` that have **at least one** primitive element with an `invalidValue`.

⌋()

**[constr_1303] Applicability of `TextTableMapping` depending on the value of `CompuMethod.category`** ⌈ If a `DataPrototypeMapping` aggregates a `TextTableMapping` then only certain combinations of the value of the applicable `CompuMethod.category` are supported:

- category of firstDataPrototype : TEXTTABLE ,
  category of secondDataPrototype : TEXTTABLE
- category of firstDataPrototype : SCALE_LINEAR_AND_TEXTTABLE ,
  category of secondDataPrototype : TEXTTABLE
- category of firstDataPrototype : TEXTTABLE ,
  category of secondDataPrototype : SCALE_LINEAR_AND_TEXTTABLE
- category of firstDataPrototype : BITFIELD_TEXTTABLE ,
  category of secondDataPrototype : TEXTTABLE
- category of firstDataPrototype : TEXTTABLE ,
  category of secondDataPrototype : BITFIELD_TEXTTABLE
- category of firstDataPrototype : BITFIELD_TEXTTABLE ,
  category of secondDataPrototype : BITFIELD_TEXTTABLE

⌋()

**[constr_1304] Existence of attribute `bitfieldTextTableMaskFirst`** ⌈ The attribute bitfieldTextTableMaskFirst shall be defined **only if** the firstDataPrototype of a DataPrototypeMapping refers to a CompuMethod that has the value of category set to BITFIELD_TEXTTABLE .

⌋()

**[constr_1305] Existence of attribute `bitfieldTextTableMaskSecond`** ⌈ The attribute bitfieldTextTableMaskSecond shall be defined **only if** the secondDataPrototype of a DataPrototypeMapping refers to a CompuMethod that has the value of category set to BITFIELD_TEXTTABLE .

⌋()

**[constr_1306] Limitation of `TextTableMapping` for `CompuMethod` s that have the value of `category` set to `BITFIELD_TEXTTABLE`** ⌈ For any TextTableMapping where both firstDataPrototype and secondDataPrototype refer to CompuMethod s that have the value of category set to BITFIELD_TEXTTABLE **and** where the attribute TextTableMapping . valuePair exists the value of attribute TextTableMapping . identicalMapping shall be set to false.

⌋()

**[constr_1307] Consistency of values and masks in `TextTableMapping`** ⌈ If a TextTableMapping element defines bit masks as bitfieldTextTableMaskFirst or bitfieldTextTableMaskSecond then all contained TextTableMapping . valuePair . firstValue s as well as all TextTableMapping . valuePair . secondValue s shall **not** specify a value that would be ruled out when - depending on the given value of TextTableMapping . mappingDirection - the relevant bit mask is applied.

⌋()

**[constr_1308] Existence of `NvBlockNeeds . cyclicWritingPeriod`** ⌈ The attribute `NvBlockNeeds . cyclicWritingPeriod` shall exist if and only if the attribute `NvBlockNeeds . storeCyclic` exists and its value is set to `true` .

⌋()

**[constr_1309] Existence of `NvBlockDescriptor . timingEvent`** ⌈ The attribute `NvBlockDescriptor . timingEvent` shall exist if and only if the `NvBlockDescriptor . nvBlockNeeds . storeCyclic` exists and is set to the value `true` .

⌋()

**[constr_1310] Existence of attributes of meta-class `NvBlockNeeds`** ⌈ If in the context of an `ApplicationSwComponentType` the attribute `SwcServiceDependency . serviceNeeds` is implemented by an `NvBlockNeeds` then the following attributes

- `NvBlockNeeds . storeCyclic`
- `NvBlockNeeds . cyclicWritingPeriod`
- `NvBlockNeeds . storeEmergency`
- `NvBlockNeeds . storeImmediate`

shall only exist if in the context of the same `SwcServiceDependency` a `SwcServiceDependency . assignedPort` exists that has the attribute `role` set to the value `NvDataPort` .

⌋()

**[constr_1311] Appearance of safety-related possible values of `MemorySection . option` or `SwAddrMethod . option`** ⌈ Any given collection of values stored in the attributes `MemorySection . option` or `SwAddrMethod . option` according to TPS_SWCT_01456 shall at most include a single value out of the following list:

- **`safetyQM`**
- **`safetyAsilA`**
- **`safetyAsilB`**
- **`safetyAsilC`**
- **`safetyAsilD`**

⌋()

**[constr_1312] `PortPrototype` s typed by a `ParameterInterface`** ⌈ `PortPrototype` s typed by a `ParameterInterface` can either be `PPortPrototype` s or `RPortPrototype` s. The usage of `PRPortPrototype` s that are typed by a `ParameterInterface` is not supported.

⌋()

**[constr_1313] Completeness of `TextTableMapping` for the values of a given bit mask on the sender side** ⌈ If a `DataPrototypeMapping` contains one or more `TextTableMapping` (s) where the `DataPrototype` on the **sender side** refers to a `CompuMethod` of `category` `BITFIELD_TEXTTABLE` then all `DataPrototypeMapping` . `textTableMapping` shall aggregate a collection of `TextTableMapping` . `valuePair` where each possible value of the **sender bit mask** Depending on the applicable case this means either `bitfieldTextTableMaskFirst` (applies if TPS_SWCT_01163 is in place) or `bitfieldTextTableMaskSecond` for the case of TPS_SWCT_01164 . is represented by exactly one `TextTableValuePair` . `firstValue` ( TPS_SWCT_01163 ) resp. `TextTableValuePair` . `secondValue` ( TPS_SWCT_01164 ).

⌋()

**[constr_1314] Profile `VSA_LINEAR` for `ApplicationArrayDataType`** ⌈ If the `dynamicArraySizeProfile` of `ApplicationArrayDataType` is set to `VSA_LINEAR` , the contained `ApplicationArrayElement` shall fulfill **all** of the following conditions:

- The attribute `ApplicationArrayElement` . `arraySizeSemantics` shall set to the value `variableSize` .
- The attribute `ApplicationArrayElement` . `maxNumberOfElements` shall be defined.
- The attribute `ApplicationArrayElement` . `arraySizeHandling` shall be set to the value `allIndicesSameArraySize` .
- The `ApplicationArrayElement` shall be typed by an `ApplicationDataType` that is not an `ApplicationArrayDataType` where the attribute `dynamicArraySizeProfile` exists.

⌋()

**[constr_1315] Profile `VSA_SQUARE` for `ApplicationArrayDataType`** ⌈ If the `dynamicArraySizeProfile` of `ApplicationArrayDataType` is set to `VSA_SQUARE` , the contained `ApplicationArrayElement` shall fulfill **all** of the following conditions:

- The attribute `ApplicationArrayElement` . `arraySizeSemantics` shall be set to the value `variableSize` .
- The attribute `ApplicationArrayElement` . `maxNumberOfElements` shall not be defined.
- The attribute `ApplicationArrayElement` . `arraySizeHandling` shall be set to the value `inheritedFromArrayElementTypeSize` .
- The `ApplicationArrayElement` shall be typed by an `ApplicationArrayDataType` .

The referred ApplicationArrayDataType shall refer over a chain (under consideration of the number of dimensions of the “root” ApplicationArrayDataType ) of nested ApplicationArrayDataType s with ApplicationArrayElement s to an ApplicationDataType that is **not** an ApplicationArrayDataType where the attribute dynamicArraySizeProfile exists.

The last ApplicationArrayDataType in that chain shall have an ApplicationArrayElement that fulfills **all** of the following conditions:

- The attribute ApplicationArrayElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ApplicationArrayElement . maxNumberOfElements shall be defined.
- The attribute ApplicationArrayElement . arraySizeHandling set to the value allIndicesSameArraySize .

All ApplicationArrayDataType s before shall have an ApplicationArrayElement that fulfills **all** of the following conditions:

- The attribute ApplicationArrayElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ApplicationArrayElement . maxNumberOfElements shall not be defined.
- The attribute ApplicationArrayElement . arraySizeHandling shall be set to the value inheritedFromArrayElementTypeSize .
- The ApplicationArrayElement shall be typed by an ApplicationArrayDataType .

⌋()

**[constr_1316] Profile VSA_RECTANGULAR for ApplicationArrayDataType** ⌈
If the dynamicArraySizeProfile of ApplicationArrayDataType is set to VSA_RECTANGULAR the contained ApplicationArrayElement shall fulfill **all** of the following conditions:

- The attribute ApplicationArrayElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ApplicationArrayElement . maxNumberOfElements shall be defined.
- The attribute ApplicationArrayElement . arraySizeHandling shall be set to the value allIndicesSameArraySize .
- The ApplicationArrayElement shall be typed by an ApplicationArrayDataType .

The referred ApplicationArrayDataType shall refer over a chain (under consideration of the number of dimensions of the “root” ApplicationArrayDataType ) of nested ApplicationArrayDataType s with ApplicationArrayElement s to an ApplicationDataType that is **not** an ApplicationArrayDataType where the attribute dynamicArraySizeProfile exists.

The last ApplicationArrayDataType in that chain shall have an ApplicationArrayElement that fulfills **all** of the following conditions:

- The attribute ApplicationArrayElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ApplicationArrayElement . maxNumberOfElements shall be defined.
- The attribute ApplicationArrayElement . arraySizeHandling shall be set to the value allIndicesSameArraySize .

All ApplicationArrayDataType s before shall have an ApplicationArrayElement that fulfills **all** of the following conditions:

- The attribute ApplicationArrayElement . arraySizeSemantics shall set to the value variableSize
- The attribute ApplicationArrayElement . maxNumberOfElements shall be defined.
- The attribute ApplicationArrayElement . arraySizeHandling shall be set to the value allIndicesSameArraySize .
- The ApplicationArrayElement shall be typed by an ApplicationArrayDataType .

⌋()

**[constr_1317] Profile VSA_FULLY_FLEXIBLE for ApplicationArrayDataType**
⌈ If the dynamicArraySizeProfile of ApplicationArrayDataType is set to VSA_FULLY_FLEXIBLE , the contained ApplicationArrayElement shall fulfill **all** of the following conditions:

- The attribute ApplicationArrayElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ApplicationArrayElement . maxNumberOfElements shall be defined.
- The attribute ApplicationArrayElement . arraySizeHandling shall be set to the value allIndicesDifferentArraySize .
- The ApplicationArrayElement shall be typed by an ApplicationArrayDataType .

The referred ApplicationArrayDataType shall refer over a chain (under consideration of the number of dimensions of the “root” ApplicationArrayDataType ) of nested ApplicationArrayDataType s with ApplicationArrayElement s to an ApplicationDataType that is **not** an ApplicationArrayDataType where the attribute dynamicArraySizeProfile exist.

The last ApplicationArrayDataType in that chain shall have an ApplicationArrayElement that fulfills **all** of the following conditions:

- The attribute ApplicationArrayElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ApplicationArrayElement . maxNumberOfElements shall be defined.
- The attribute ApplicationArrayElement . arraySizeHandling shall be set to the value allIndicesSameArraySize .

All ApplicationArrayDataType s before shall have an ApplicationArrayElement that fulfills **all** of the following conditions:

- The attribute ApplicationArrayElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ApplicationArrayElement . maxNumberOfElements shall be defined.
- The attribute ApplicationArrayElement . arraySizeHandling shall be set to the value allIndicesDifferentArraySize .
- The ApplicationArrayElement shall be typed by an ApplicationArrayDataType .

⌋()

**[constr_1318] Profile VSA_LINEAR for ImplementationDataType** ⌈ If the value of attribute ImplementationDataType . dynamicArraySizeProfile is set to VSA_LINEAR , the ImplementationDataType shall aggregate a VSA Payload ImplementationDataTypeElement that fulfills all of the following conditions:

- The attribute ImplementationDataTypeElement . arraySizeSemantics shall not be defined.
- The attribute ImplementationDataTypeElement . category shall be set to ARRAY .
- The attribute ImplementationDataTypeElement . arraySize shall not be defined.
- The attribute ImplementationDataTypeElement . arraySizeHandling shall not be defined.

The VSA Payload ImplementationDataTypeElement shall immediately aggregate another ImplementationDataTypeElement that shall fulfill all of the following conditions:

- The attribute ImplementationDataTypeElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ImplementationDataTypeElement . arraySize shall be defined.
- The attribute ImplementationDataTypeElement . arraySizeHandling shall be set to the value allIndicesSameArraySize .

⌋()

**[constr_1319] Profile VSA_SQUARE for ImplementationDataType** ⌈ If the value of attribute ImplementationDataType . dynamicArraySizeProfile is set to VSA_SQUARE , the ImplementationDataType shall aggregate a VSA Payload ImplementationDataTypeElement that fulfills all of the the following conditions:

- The attribute ImplementationDataTypeElement . arraySizeSemantics shall not be defined.
- The attribute ImplementationDataTypeElement . category shall be set to the value ARRAY .
- The attribute ImplementationDataTypeElement . arraySize shall not be defined.
- The attribute ImplementationDataTypeElement . arraySizeHandling shall not be defined.

The VSA Payload ImplementationDataTypeElement shall immediately aggregate another ImplementationDataTypeElement (representing the first dimension) that shall fulfill all of the following conditions:

- The attribute ImplementationDataTypeElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ImplementationDataTypeElement . category shall be set to the value ARRAY .
- The attribute ImplementationDataTypeElement . arraySize shall not be defined.
- The attribute ImplementationDataTypeElement . arraySizeHandling shall be set to the value inheritedFromArrayElementTypeSize .

All **intermediate** ImplementationDataTypeElement s in the aggregation chain that do not terminate the chain shall fulfill all of the following conditions:

- The attribute ImplementationDataTypeElement . arraySizeSemantics shall be set to the value variableSize .

- The attribute ImplementationDataTypeElement . category shall be set to the value ARRAY .
- The attribute ImplementationDataTypeElement . arraySize shall not be defined.
- The attribute ImplementationDataTypeElement . arraySizeHandling shall be set to the value inheritedFromArrayElementTypeSize .

The **terminating** ImplementationDataTypeElement in the aggregation chain shall fulfill all of the following conditions:

- The attribute ImplementationDataTypeElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ImplementationDataTypeElement . arraySize shall be defined.
- The attribute ImplementationDataTypeElement . arraySizeHandling shall be set to the value allIndicesSameArraySize .

⌋()

**[constr_1320] Profile VSA_RECTANGULAR for ImplementationDataType** ⌈ If the value of attribute ImplementationDataType . dynamicArraySizeProfile is set to VSA_RECTANGULAR , the ImplementationDataType shall aggregate a VSA Payload ImplementationDataTypeElement that fulfills all of the following conditions:

- The attribute ImplementationDataTypeElement . arraySizeSemantics shall not be defined.
- The attribute ImplementationDataTypeElement . category shall be set to the value ARRAY .
- The attribute ImplementationDataTypeElement . arraySize shall not be defined.
- The attribute ImplementationDataTypeElement . arraySizeHandling shall not be defined.

The VSA Payload ImplementationDataTypeElement shall immediately aggregate another ImplementationDataTypeElement (representing the first dimension) that shall fulfill all of the following conditions:

- The attribute ImplementationDataTypeElement . category shall be set to the value ARRAY .
- The attribute ImplementationDataTypeElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ImplementationDataTypeElement . arraySize shall be defined.

- The attribute ImplementationDataTypeElement . arraySizeHandling shall be set to the value allIndicesSameArraySize .

All **intermediate** ImplementationDataTypeElement s in the aggregation chain that do not terminate the chain shall fulfill all of the following conditions:

- The attribute ImplementationDataTypeElement . category shall be set to the value ARRAY .
- The attribute ImplementationDataTypeElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ImplementationDataTypeElement . arraySize shall be defined.
- The attribute ImplementationDataTypeElement . arraySizeHandling shall be set to the value allIndicesSameArraySize .

The **terminating** ImplementationDataTypeElement in the aggregation chain shall fulfill all of the following conditions:

- The attribute ImplementationDataTypeElement . arraySizeSemantics shall be set to the value variableSize .
- The attribute ImplementationDataTypeElement . arraySize shall be defined.
- The attribute ImplementationDataTypeElement . arraySizeHandling shall be set to the value allIndicesSameArraySize .

⌋()

**[constr_1321] Profile VSA_FULLY_FLEXIBLE for ImplementationDataType** ⌈
If the value of attribute ImplementationDataType . dynamicArraySizeProfile is set to the value VSA_FULLY_FLEXIBLE , the ImplementationDataType shall aggregate a VSA Payload ImplementationDataTypeElement that fulfills all of the following conditions:

- The attribute ImplementationDataTypeElement . arraySizeSemantics shall not be defined.
- The attribute ImplementationDataTypeElement . category shall be set to the value ARRAY .
- The attribute ImplementationDataTypeElement . arraySize shall not be defined
- The attribute ImplementationDataTypeElement . arraySizeHandling shall not be defined.

The VSA Payload ImplementationDataTypeElement shall immediately aggregate another ImplementationDataTypeElement (representing the first dimension) that shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.category` shall be set to STRUCTURE
- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesDifferentArraySize`.

The `ImplementationDataTypeElement` shall aggregate another `ImplementationDataTypeElement` that fulfills the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall not be defined.
- The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySize` shall not be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall not be defined.

The **aggregation chain is continued** by a (possible empty) sequence of a pair of `ImplementationDataTypeElement`s with the following characteristics:

- The first `ImplementationDataTypeElement` in the pair shall fulfill all of the following conditions:
  - The attribute `ImplementationDataTypeElement.category` shall be set to `STRUCTURE`.
  - The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
  - The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
  - The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesDifferentArraySize`.
- The second `ImplementationDataTypeElement` in the pair shall fulfill all of the following conditions:
  - The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall not be defined.
  - The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.

  - The attribute `ImplementationDataTypeElement.arraySize` shall not be defined.
  - The attribute `ImplementationDataTypeElement.arraySizeHandling` shall not be defined.

The **terminating** `ImplementationDataTypeElement` in the aggregation chain shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.

⌋()

**[constr_1322] `Size Indicator` for undefined `dynamicArraySizeProfile`** ⌈ If the `ImplementationDataType.dynamicArraySizeProfile` does not exists but the `ImplementationDataType` is mapped to an `ApplicationArrayDataType` where the attribute `ApplicationArrayDataType.dynamicArraySizeProfile` exists, then the `ImplementationDataType` shall have the `category STRUCTURE`, representing a `Variable-Size Array Data Type` with `Size Indicator` enabled.

⌋()

**[constr_1363] Existence of attributes of `DiagnosticValueNeeds`** ⌈ if `DiagnosticValueNeeds` is aggregated by a `SwcServiceDependency` in the role `serviceNeeds` then the attributes

- `DiagnosticValueNeeds.diagnosticValueAccess`
- `DiagnosticValueNeeds.dataLength`

shall **not** exist.

⌋()

**[constr_1364] Existence of attributes of `DiagnosticIoControlNeeds`** ⌈ if `DiagnosticIoControlNeeds` is aggregated by a `SwcServiceDependency` in the role `serviceNeeds` then the attributes

- `DiagnosticIoControlNeeds.freezeCurrentStateSupported`
- `DiagnosticIoControlNeeds.shortTermAdjustmentSupported`

shall **not** exist.

⌋()

**[constr_1375] Existence of attributes of `CompuMethod` and related meta-classes** ⌈ The existence of attributes of `CompuMethod` and related meta-classes depending on the value of the `category` shall follow the restrictions documented in Table `table_3a_CategoriesCompuMethod`.

⌋()

**[constr_1381] Appearance of core-related possible values of `MemorySection.option` or `SwAddrMethod.option`** ⌈ Any given collection of values stored in the attributes `MemorySection.option` or `SwAddrMethod.option` according to TPS_SWCT_01456 shall at most include a single value out of the following list:

- **`coreGlobal`**
- **`coreLocal`**

⌋()

**[constr_1382] Mutually exclusive existence of attributes `SwVariableRefProxy.autosarVariable` vs. `SwVariableRefProxy.mcDataInstanceVar`** ⌈ In any given AUTOSAR model, the aggregations `SwVariableRefProxy.autosarVariable` and `SwVariableRefProxy.mcDataInstanceVar` shall never exist at the same time.

⌋()

**[constr_1383] Existence of `CompuMethod` and `DataConstr` for `ImplementationDataType`s of `category` `TYPE_REFERENCE`** ⌈ The existence of `ImplementationDataType.swDataDefProps.compuMethod` and `ImplementationDataType.swDataDefProps.dataConstr` for `ImplementationDataType`s of `category` `TYPE_REFERENCE` is only allowed if the respective `ImplementationDataType`, after all type references are resolved, ends up in an `ImplementationDataType` of `category` `VALUE`.

⌋()

**[constr_1384] Definition of `invalidValue` for `DataPrototype` typed by `ApplicationPrimitiveDataType` of `category` `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, `RES_AXIS`, and `VAL_BLK`** ⌈ An `invalidValue` **shall not** be specified for a `DataPrototype` typed by `ApplicationPrimitiveDataType` of `category` `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, `RES_AXIS`, and `VAL_BLK`

⌋()

**[constr_1385] `DataPrototype` is typed by an `ImplementationDataType`** ⌈ If a `DataPrototype` is typed by an `ImplementationDataType` its `initValue` shall **not** be provided by an `ApplicationValueSpecification`.

⌋()

**[constr_1386] `PortDefinedArgumentValue` shall only be defined for `AbstractProvidedPortPrototype`** ⌈ A `PortAPIOption` which aggregates at least one `PortDefinedArgumentValue` in the role `portArgValue` shall reference an `AbstractProvidedPortPrototype` typed by a `ClientServerInterface` in the role `port` .

⌋()

**[constr_1388] `VariationPointProxy` of `category` `VALUE` shall not mix "pre-build" and "post-build" use-cases** ⌈ If the value of `category` of the `VariationPointProxy` is set to `VALUE` then there can only be one value yield from the evaluation of a `VariationPointProxy` . In other words, a `VariationPointProxy` of `category` `VALUE` shall not mix the "pre-build" and "post-build" use-cases.

⌋()

**[constr_1389] Restriction regarding the value of `category` of `VariationPointProxy` . `implementationDataType`** ⌈ `VariationPointProxy` . `implementationDataType` shall **not** be of `category` `STRUCTURE` , `ARRAY` , `UNION` , `FUNCTION_REFERENCE` , and `DATA_REFERENCE` .

The `VariationPointProxy` . `implementationDataType` shall be of `category` `VALUE` or `TYPE_REFERENCE` that, after all references are resolved, yields an `ImplementationDataType` of `category` `VALUE` .

⌋()

**[constr_1390] Restriction to the value of `SenderReceiverInterface` . `invalidationPolicy` . `handleInvalid`** ⌈ If the value of `SenderReceiverInterface` . `invalidationPolicy` . `handleInvalid` is set to any value other than `HandleInvalidEnum` . `dontInvalidate` then the `invalidValue` shall not be within the interval defined by the `CompuMethod` of the applicable `dataElement` .

⌋()

**[constr_1391] Compatibility of `Unit` s in the context of assignment using an `ApplicationValueSpecification`** ⌈ If an `ApplicationValueSpecification` is used in the context of an assignment to an `AutosarDataPrototype` then the `ApplicationValueSpecification` . `swValueCont` . `unit` shall be compatible to the `Unit` used in the definition of the given `AutosarDataPrototype` , i.e. `AutosarDataType` . `swDataDefProps` . `unit` .

⌋()

**[constr_1392] Compatibility of `Unit` s in the context of assignment using an `ApplicationRuleBasedValueSpecification`** ⌈ If an `ApplicationRuleBasedValueSpecification` is used in the context of an assignment to an `AutosarDataPrototype` then the `ApplicationRuleBasedValueSpecification` . `swValueCont` . `unit` shall be compatible to the `Unit` used in the definition of the given `AutosarDataPrototype` , i.e. `AutosarDataType` . `swDataDefProps` . `unit` .

⌋()

**[constr_1393] Existence of `RuleBasedValueCont . unit`** ⌈ For every `RuleBasedValueCont` the attribute `unit` shall exist.

⌋()

**[constr_1395] `NvBlockDataMapping` shall be complete** ⌈ If an `NvBlockDataMapping` refers to *sub-elements* or *leaf* elements of the `NvDataInterface . nvData` in the context of a particular `PortPrototype` then **all remaining** *sub-elements* or *leaf* elements **shall effectively be mapped** according to TPS_SWCT_01659 by means of a collection of `NvBlockDataMapping` s.

⌋()

**[constr_1396] Restriction for the value of attribute `category` for non-terminating `ImplementationDataTypeElement` s taken to model a `Variable-Size Array Data Type`** ⌈ The value of attribute `category` for non-terminating `ImplementationDataTypeElement` s taken to model a `Variable-Size Array Data Type` shall **not** be set to `TYPE_REFERENCE` .

⌋()

**[constr_1397] Existence of attributes of `TransformerHardErrorEvent`** ⌈ For any given `TransformerHardErrorEvent` , **either** the attribute `TransformerHardErrorEvent . operation` **or** `TransformerHardErrorEvent . trigger` shall exist.

⌋()

**[constr_1398] Existence of attributes of `BaseTypeDirectDefinition`** ⌈ If the value of attribute `BaseTypeDirectDefinition . baseTypeEncoding` is set to `UTF-16` then the attribute `BaseTypeDirectDefinition . byteOrder` shall exist.

The only allowed values of `BaseTypeDirectDefinition . byteOrder` in this case are `mostSignificantByteFirst` and `mostSignificantByteLast`

⌋()

**[constr_1399] Standardized values of `ModeDeclarationGroup . category`** ⌈ The AUTOSAR standard defines the following values of the attribute `ModeDeclarationGroup . category` with a standardized meaning:

- `EXPLICIT_ORDER`
- `ALPHABETIC_ORDER`

TPS_SWCT_01010 defines the meaning of these values.

It is **not allowed** to define any custom or project-specific value of the attribute `ModeDeclarationGroup . category` .

⌋()

**[constr_1400] Reference to a specific `DataTransformation`** ⌈ A specific `DataTransformation` shall only be referenced by either

- a DataPrototypeMapping in the role firstToSecondDataTransformation **or**
- an ISignal in the role dataTransformation **or**
- an ISignalGroup in the role comBasedSignalGroupTransformation **or**
- a ClientServerOperationMapping in the role firstToSecondDataTransformation

⌋()

**[constr_1401] Restrictions on the relation between DataPrototypeMapping and DataTransformation** ⌈ A VariableDataPrototype in the context of a PortPrototype shall **not** be referenced by a DataPrototypeMapping that references a DataTransformation while a DataMapping exists that points to this VariableDataPrototype (via the SystemSignal ) that also refers to an ISignal that in turn references a DataTransformation .

⌋()

**[constr_1402] Applicability of core-related possible values of MemorySection . option or SwAddrMethod . option related to SwAddrMethod . sectionInitializationPolicy** ⌈ If the attribute SwAddrMethod . option or MemorySection . option is set to coreLocal then the attribute SwAddrMethod . sectionInitializationPolicy of the same SwAddrMethod respectively the MemorySection . swAddrmethod shall be either set to INIT or CLEARED .

⌋()

**[constr_1403] NvBlockDataMapping s to a given nvData shall be unambiguous** ⌈ If an NvBlockDataMapping exists that **directly** and **completely** maps a specific NvDataInterface . nvData in the context of a particular PortPrototype then **no** other NvBlockDataMapping which maps sub-elements of the NvDataInterface . nvData shall exist.

⌋()

**[constr_1404] All NvDataInterface . nvData of PortPrototype s in the context of a specific SwcServiceDependency shall be mapped to the same NvBlockDescriptor** ⌈ In the context of a given SwcServiceDependency (which, in turn, is owned by an AtomicSwComponentType ), **all** NvDataInterface . nvData of PortPrototype s referenced by a RoleBasedPortAssignment with attribute RoleBasedPortAssignment . role set to NvDataPort shall be connected (either directly or via the definition of suitable PortInterfaceMapping s) to NvDataInterface . nvData (on the side of the NvBlockSwComponentType ) that are **completely mapped** (via NvBlockDataMapping s) **to the identical** NvBlockDescriptor . ramBlock .

⌋()

**[constr_1407] Definition of `SwDataDefProps . dataConstr` depending on the capabilities of the data type** ⌈ The definition of a `SwDataDefProps . dataConstr` according to constr_1288 and constr_1289 is only supported for a `DataPrototype` of `category ARRAY` if the corresponding `ApplicationArrayDataType` resp. `ImplementationDataType` of `category ARRAY` also supports the specification of a `SwDataDefProps . dataConstr .`

⌋()

**[constr_1408] Definition of `SwDataDefProps . displayFormat` depending on the capabilities of the data type** ⌈ The definition of a `SwDataDefProps . displayFormat` according to constr_1288 and constr_1289 is only supported for a `DataPrototype` of `category ARRAY` if the corresponding `ApplicationArrayDataType` resp. `ImplementationDataType` of `category ARRAY` also supports the specification of a `SwDataDefProps . displayFormat .`

⌋()

**[constr_1409] Definition of `SwDataDefProps . dataConstr` depending on the capabilities of the element data type** ⌈ The definition of a `SwDataDefProps . dataConstr` according to constr_1007 and constr_1009 is only supported for an `ApplicationArrayDataType` resp. an `ImplementationDataType` of `category ARRAY` if the aggregated `ApplicationArrayDataType . element` resp. `ImplementationDataType . subElement` also supports the specification of a `SwDataDefProps . dataConstr .`

⌋()

**[constr_1410] Definition of `SwDataDefProps . displayFormat` depending on the capabilities of the element data type** ⌈ The definition of a `SwDataDefProps . displayFormat` according to constr_1007 and constr_1009 is only supported for an `ApplicationArrayDataType` resp. an `ImplementationDataType` of `category ARRAY` if the aggregated `ApplicationArrayDataType . element` resp. `ImplementationDataType . subElement` also supports the specification of a `SwDataDefProps . displayFormat .`

⌋()

**[constr_1413] Definition of `SwDataDefProps . stepSize` depending on the capabilities of the data type** ⌈ The definition of a `SwDataDefProps . stepSize` according to constr_1288 and constr_1289 is only supported for a `DataPrototype` of `category ARRAY` if the corresponding `ApplicationArrayDataType` resp. `ImplementationDataType` of `category ARRAY` also supports the specification of a `SwDataDefProps . stepSize .`

⌋()

**[constr_1414] Definition of `SwDataDefProps . stepSize` depending on the capabilities of the element data type** ⌈ The definition of a `SwDataDefProps . stepSize` according to constr_1007 and constr_1009 is only supported for an `ApplicationArrayDataType` resp. an `ImplementationDataType` of `category ARRAY`

if the aggregated ApplicationArrayDataType . element resp. ImplementationDataType . subElement also supports the specification of a SwDataDefProps . stepSize .

⌋()

**[constr_1415] Supported values of ModeSwitchEventTriggeredActivity . role** ⌈ The only supported value of ModeSwitchEventTriggeredActivity . role is WriteBlock .

⌋()

**[constr_1416] Existence of ApplicationArrayElement . maxNumberOfElements** ⌈ The attribute ApplicationArrayElement . maxNumberOfElements shall exist for all ApplicationArrayElement s defined in the scope of an ApplicationArrayDataType where the attribute ApplicationArrayDataType . dynamicArraySizeProfile does not exist.

⌋()

**[constr_1417] Invalid connection between NvBlockSwComponentType and other AtomicSwComponentType (I)** ⌈ A configuration where an RPortPrototype owned by an AtomicSwComponentType is simultaneously and directly connected to AbstractProvidedPortPrototype s of a collection of AtomicSwComponentType s where at least one in the collection is an NvBlockSwComponentType for a matching set of dataElement s in all these PortPrototype s shall be considered invalid.

⌋()

**[constr_1418] Invalid connection between NvBlockSwComponentType and other AtomicSwComponentType (II)** ⌈ A configuration where a PRPortPrototype owned by an AtomicSwComponentType is connected to a PPortPrototype owned by an NvBlockSwComponentType for a matching set of dataElement s in all these PortPrototype s shall be considered invalid.

⌋()

**[constr_1420] Existence of SwAxisIndividual . inputVariableType** ⌈ If the reference SwAxisIndividual . inputVariableType does not exist then either:

- SwAxisIndividual . dataConstr
- SwAxisIndividual . unit

or

- SwAxisIndividual . dataConstr
- SwAxisIndividual . compuMethod . unit

shall exist.

⌋()

**[constr_1422] Value of `category` is `VOID`** ⌈ If the value of the attribute `SwBaseType.category` is set to `VOID` then the attribute `baseTypeSize` shall not exist.

⌋()

**[constr_1423] Completeness of references `ArVariableInImplementationDataInstanceRef.contextDataPrototype`** ⌈ The reference `ArVariableInImplementationDataInstanceRef.contextDataPrototype` shall be defined for

- each *leaf* (i.e. the end of a chain of aggregating elements) `ImplementationDataTypeElement` of `category` `TYPE_REFERENCE` in a chain of referencing `ImplementationDataType` s which is not the `targetDataPrototype`
- and each `ImplementationDataTypeElement` owned by an `ImplementationDataType` or `ImplementationDataTypeElement` of `category` `ARRAY` in a chain of referencing `ImplementationDataType` s

starting from the `ImplementationDataType` s of the `rootVariableDataPrototype` down to the leaf `ImplementationDataTypeElement` which is typed (directly or indirectly via `ImplementationDataType` of `category` `TYPE_REFERENCE` ) by the `ImplementationDataType` of the `targetDataPrototype` .

⌋()

**[constr_1424] Existence of `ArVariableInImplementationDataInstanceRef.contextDataPrototype`** ⌈ The attribute `ArVariableInImplementationDataInstanceRef.contextDataPrototype` shall only exist for an `ImplementationDataTypeElement` `category` `TYPE_REFERENCE` or `ARRAY` .

⌋()

**[constr_1425] Definition of `swCalprmAxisSet.swCalprmAxis` / `SwAxisIndividual.swVariableRef` depending on the capabilities of the data type** ⌈ The definition of a `swCalprmAxisSet.swCalprmAxis` / `SwAxisIndividual.swVariableRef` in the context of an `InstantiationDataDefProps` or a `ParameterAccess` is only supported for a `DataPrototype` of `category` `ARRAY` if the data type of the `ApplicationArrayElement` also supports the specification of a `swCalprmAxisSet.swCalprmAxis` / `SwAxisIndividual.swVariableRef` according to constr_1289 .

Thereby, multiple `ApplicationArrayDataType` s might be nested to express multiple array dimensions.

⌋()

**[constr_1426] Consistency of array sizes for axes and input variable array** ⌈ The number of array dimension defined by `ApplicationArrayDataType` s and the values of the `maxNumberOfElements` attributes for the array of elements of `category` `CURVE` , `MAP` , `CUBOID` , `CUBE_4` , `CUBE_5` , `COM_AXIS` , or `RES_AXIS` shall be **identical** to the number of array dimension and according value of the `maxNumberO-`

fElements of the VariableDataPrototype referenced by SwAxisIndividual . swVariableRef . autosarVariable .

⌋()

**[constr_1427] Definition of swCalprmAxisSet . swCalprmAxis / SwAxisGrouped . swCalprmRef depending on the capabilities of the data type** ⌈ The definition of a swCalprmAxisSet . swCalprmAxis / SwAxisGrouped . swCalprmRef in the context of an InstantiationDataDefProps or a ParameterAccess is only supported for a DataPrototype of category ARRAY if the data type of the ApplicationArrayElement also supports the specification of a swCalprmAxisSet . swCalprmAxis / SwAxisGrouped . swCalprmRef according to constr_1289 .

Thereby, multiple ApplicationArrayDataType s might be nested to express multiple array dimensions.

⌋()

**[constr_1428] Consistency of array sizes for arrays of elements of category CURVE , MAP , CUBOID , CUBE_4 , or CUBE_5 arrays and used group axes arrays** ⌈ The number of array dimension defined by ApplicationArrayDataType s and the values of attribute maxNumberOfElements attributes for the array of elements of category CURVE , MAP , CUBOID , CUBE_4 , or CUBE_5 needs to be identical to the number of array dimension and according value of the maxNumberOfElements of the DataPrototype referenced by SwAxisGrouped . swCalprmRef . arParameter .

⌋()

**[constr_1429] Access to data within PortPrototype s from within RunnableEntity s** ⌈ For a VariableAccess that is aggregated in the roles

- RunnableEntity . dataWriteAccess
- RunnableEntity . dataReadAccess
- RunnableEntity . dataSendPoint
- RunnableEntity . dataReceivePointByArgument
- RunnableEntity . dataReceivePointByValue

the existence of the following attributes is not allowed:

- VariableAccess . accessedVariable . autosarVariable . contextDataPrototype
- VariableAccess . accessedVariable . autosarVariable . rootVariableDataPrototype
- VariableAccess . accessedVariable . autosarVariableInImplDatatype
- VariableAccess . accessedVariable . localVariable

In other words: in this case, only the references VariableAccess . accessedVariable . autosarVariable . portPrototype and VariableAccess . accessedVariable . autosarVariable . targetDataPrototype shall exist and the latter shall **exclusively** refer to a VariableDataPrototype that is aggregated as either

- SenderReceiverInterface . dataElement or
- NvDataInterface . nvData .

⌋()

**[constr_1430] Access to local data from within RunnableEntity s** ⌈ For VariableAccess that is aggregated in the roles

- RunnableEntity . writtenLocalVariable
- RunnableEntity . readLocalVariable

the existence of the following attributes is not allowed:

- VariableAccess . accessedVariable . autosarVariableInImplDatatype
- VariableAccess . accessedVariable . autosarVariable

In other words, **only** the reference VariableAccess . accessedVariable . localVariable shall be used in this case.

⌋()

**[constr_1431] Access to parameters from within RunnableEntity s** ⌈ For a ParameterAccess that is aggregated in the role RunnableEntity . parameterAccess the existence of the following attributes is not allowed:

- ParameterAccess . accessedParameter . autosarParameter . contextDataPrototype
- ParameterAccess . accessedParameter . autosarParameter . rootParameterDataPrototype

In other words: in this case, **one of** of the following alternatives is allowed to exist:

- a combination of
  - ParameterAccess . accessedParameter . autosarParameter . portPrototype and
  - ParameterAccess . accessedParameter . autosarParameter . targetDataPrototype that **exclusively** refers to a ParameterDataPrototype aggregated by a ParameterInterface in the role parameter .
- ParameterAccess . accessedParameter . localParameter that refers to a ParameterDataPrototype that is either aggregated as

- InternalBehavior . constantMemory or
- SwcInternalBehavior . perInstanceParameter or
- SwcInternalBehavior . sharedParameter .

⌋()

**[constr_1432] Multiplicity of CommunicationBufferLocking** ⌈ In a concrete aggregated set of PortAPIOption . supportedFeature , CommunicationBufferLocking shall exist **at most once** .

⌋()

**[constr_1433] Transient faults are not applicable to software-components** ⌈ An ErrorTracerNeeds aggregated in the context of a SwcInternalBehavior is not allowed to own a TransientFault in the role ErrorTracerNeeds . tracedFailure .

⌋()

**[constr_1434] CompuScale s shall not have identical CompuScale Value Symbolic Name s** ⌈ In a CompuMethod that is subject to constr_1146 , no two CompuScale s shall have identical CompuScale Value Symbolic Name s (according to TPS_SWCT_01696 ).

⌋()

**[constr_1438] ApplicationArrayElement . indexDataType needs to refer to a CompuMethod of category TEXTTABLE** ⌈ The reference ApplicationArrayElement . indexDataType shall only point to an ApplicationPrimitiveDataType that in turn refers to a CompuMethod of category TEXTTABLE .

⌋()

**[constr_1439] Requirements on ApplicationArrayElement if attribute indexDataType exists** ⌈ If ApplicationArrayElement . indexDataType exists then the attribute ApplicationArrayElement . arraySizeSemantics shall be set to the value fixedSize and attribute arraySizeHandling shall not exist.

⌋()

**[constr_1440] Size of the CompuMethod of category TEXTTABLE referenced by ApplicationArrayElement . indexDataType** ⌈ The interval defined by the CompuScale s contained in the CompuMethod referenced by ApplicationArrayElement . indexDataType shall start at 0 and include all integer values until ApplicationArrayElement . maxNumberOfElements - 1.

⌋()

**[constr_1442] category TYPE_REFERENCE shall not be used for modeling the "payload" of a Wrapped Union Data Type** ⌈ For the modeling of the "payload" part of a Wrapped Union Data Type it shall not be possible to use an Implemen-

tationDataTypeElement of category TYPE_REFERENCE that finally (i.e. after all possible indirections are resolved) boils down to category UNION .

⌋()

**[constr_1443] category UNION shall not be used for ImplementationDataType** ⌈ The value UNION of category shall **not** be used for an ImplementationDataType .

⌋()

**[constr_1444] Limited applicability of Wrapped Union Data Type** ⌈ There is no support for the usage of Wrapped Union Data Type in ramBlock , romBlock , PerInstanceMemory , PortInterfaceMapping s, and Diagnostics.

⌋()

**[constr_1445] Initialization of the Member Selector of a Wrapped Union Data Type** ⌈ The initValue for the Member Selector shall **never be set to any value other than 1** .

⌋()

**[constr_1446] No definition of invalidValue for a Wrapped Union Data Type** ⌈ The definition of an invalidValue for a DataPrototype typed by a Wrapped Union Data Type is not supported.

⌋()

**[constr_1468] Limitation on the number of SwcExclusiveAreaPolicy s** ⌈ An ExclusiveArea shall only be referenced by **at most** one SwcExclusiveAreaPolicy .

⌋()

**[constr_1469] Applicability of constraints depending on the existence of a data transformation** ⌈ constr_1269 , constr_1270 , constr_1268 , and constr_1240 shall **not** apply under the following conditions:

- A reference from the respective ClientServerOperationMapping to a DataTransformation in the role firstToSecondDataTransformation exists.
- The value of the attribute dataTransformationKind of the referenced DataTransformation is set to DataTransformationKindEnum . asymmetricFromByteArray or DataTransformationKindEnum . asymmetricToByteArray .

⌋()

**[constr_1515] Existence of ImplementationDataTypeSubElementRef . implementationDataTypeElement as opposed to ImplementationDataType-**

**SubElementRef . parameterImplementationDataTypeElement** ⌈ For any given ImplementationDataTypeSubElementRef , either the aggregation

- ImplementationDataTypeSubElementRef . implementationDataTypeElement or
- ImplementationDataTypeSubElementRef . parameterImplementationDataTypeElement

shall exist.

⌋()

**[constr_1516] Completeness of references ArParameterInImplementationDataInstanceRef . contextDataPrototype** ⌈ The reference ArParameterInImplementationDataInstanceRef . contextDataPrototype shall be defined for

- each *leaf* (i.e. the end of a chain of aggregating elements) ImplementationDataTypeElement of category TYPE_REFERENCE in a chain of referencing ImplementationDataType s which is not the targetDataPrototype
- and each ImplementationDataTypeElement owned by an ImplementationDataType or ImplementationDataTypeElement of category ARRAY in a chain of referencing ImplementationDataType s

starting from the ImplementationDataType s of the rootParameterDataPrototype down to the leaf ImplementationDataTypeElement which is typed (directly or indirectly via ImplementationDataType of category TYPE_REFERENCE ) by the ImplementationDataType of the targetDataPrototype .

⌋()

**[constr_1517] Existence of ArParameterInImplementationDataInstanceRef . contextDataPrototype** ⌈ The attribute ArParameterInImplementationDataInstanceRef . contextDataPrototype shall only exist for an ImplementationDataTypeElement category TYPE_REFERENCE or ARRAY .

⌋()

**[constr_1518] Consistency of data types in the context of ArParameterInImplementationDataInstanceRef** ⌈ The definition of attributes contextDataPrototype and targetDataPrototype shall be enclosed in the context of the definition of the data type used to type rootParameterDataPrototype .

⌋()

**[constr_1519] Existence of attributes vs. category of ApplicationValueSpecification** ⌈ The existence of attributes of meta-class ApplicationValueSpecification vs. the value of category is regulated by Table fig_3a_AUTOSAR_Meta_2d_Model_Model_DOC_ApplicationValueSpecification .

⌋()

**[constr_1520] Semantics of `ObdRatioServiceNeeds.rateBasedMonitoredEvent`** ⌈ In the context of an `SwcServiceDependency`, each `DiagnosticEventNeeds` referenced in the role `rateBasedMonitoredEvent` shall only be referenced by at most a single `ObdRatioServiceNeeds`.

⌋()

**[constr_1521] Reference from `AsynchronousServerCallReturnsEvent` to `AsynchronousServerCallResultPoint`** ⌈ In the context of a `RunnableEntity`, a given `AsynchronousServerCallResultPoint` shall only be referenced by one `AsynchronousServerCallReturnsEvent` in the role `eventSource`.

⌋()

**[constr_1523] No mode disabling for `OperationInvokedEvents`** ⌈ An `OperationInvokedEvent` shall not have a reference to a `ModeDeclaration` in the role `disabledMode`.

⌋()

**[constr_1538] Restriction for `ReceiverComSpec.dataElement`** ⌈ The reference `ReceiverComSpec.dataElement` **shall not** refer to an `ArgumentDataPrototype` or `ParameterDataPrototype`.

⌋()

**[constr_1539] Restriction for `SenderComSpec.dataElement`** ⌈ The reference `SenderComSpec.dataElement` **shall not** refer to an `ArgumentDataPrototype` or `ParameterDataPrototype`.

⌋()

**[constr_1540] Existence of `ClientComSpec.operation`** ⌈ The reference `ClientComSpec.operation` **shall exist** if the `AbstractRequiredPortPrototype` that owns the `ClientComSpec` is typed by a `ClientServerInterface`.

⌋()

**[constr_1541] Existence of `ServerComSpec.operation`** ⌈ The reference `ServerComSpec.operation` **shall exist** if the `AbstractProvidedPortPrototype` that owns the `ServerComSpec` is typed by a `ClientServerInterface`.

⌋()

**[constr_1544] Modeling of `SwAxisGeneric` for the definition of a fix axis** ⌈ The standardized values and multiplicities within the model of an `SwAxisGeneric` according to TPS_SWCT_01479 and TPS_SWCT_01480 are documented in Table `table_3a_ModelingOfSwAxisGeneric`.

⌋()

**[constr_1545] No initialization for fix axis** ⌈ An `ApplicationValueSpecification` taken to initialize an `ApplicationPrimitiveDataType` that contains a fix axis shall not contain initial values for the axis index of the fix axis inside the `ApplicationPrimitiveDataType` .

⌋()

**[constr_2000] Compatibility of `ClientServerOperation` s triggering the same `RunnableEntity`** ⌈ The `ClientServerOperation` s are considered compatible if the number of arguments (which can be `ArgumentDataPrototype` s or related `PortDefinedArgumentValue` s) is equal and the corresponding arguments (i.e. first argument on both sides, second argument on both sides, etc.) are compatible.

In particular, this means that:

- for combinations of `ArgumentDataPrototype` s and `ArgumentDataPrototype` s where the `serverArgumentImplPolicy` is set to `useArgumentType` the referred `ImplementationDataType` s shall be compatible.

  In case of data types of `category STRUCTURE` all by order matching `ImplementationDataTypeElement` s shall be named equally.

- for combinations of `PortDefinedArgumentValue` s and `ArgumentDataPrototype` s where the `serverArgumentImplPolicy` is set to `useArgumentType` the referred `ImplementationDataType` s shall be compatible.

- for combinations of `ArgumentDataPrototype` s and `ArgumentDataPrototype` s where the `serverArgumentImplPolicy` is set to `useArrayBaseType` the referred `ImplementationDataType` s of `category ARRAY` shall have compatible `ImplementationDataTypeElement` s.

  In case of `ImplementationDataTypeElement` s of `category STRUCTURE` all by order matching `ImplementationDataTypeElement` s of the structure shall be named equally.

- for `ArgumentDataPrototype` s where the `serverArgumentImplPolicy` is set to `useVoid` an arbitrary `ImplementationDataType` is referred to.

In addition, it is required that the **return value defined on both sides shall match** (in terms of `Std_ReturnType` vs. `void` ) and also the `possibleError` s are compatible.

⌋()

**[constr_2002] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `dataReadAccess`** ⌈ A `VariableAccess` in the role `dataReadAccess` shall refer to an `RPortPrototype` or `PRPortPrototype` that is typed by either a `SenderReceiverInterface` or a `NvDataInterface` .

⌋()

**[constr_2003] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `dataWriteAccess`** ⌈ A `VariableAccess` in the role `dataWriteAccess` shall refer to a `PPortPrototype` or `PRPortPrototype` that is typed by either a `SenderReceiverInterface` or a `NvDataInterface` .

⌋()

**[constr_2004] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `dataSendPoint`** ⌈ A `VariableAccess` in the role `dataSendPoint` shall refer to a `PPortPrototype` or `PRPortPrototype` that is typed by either a `SenderReceiverInterface` or a `NvDataInterface` .

⌋()

**[constr_2005] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `dataReceivePointByValue` or `dataReceivePointByArgument`** ⌈ A `VariableAccess` in the role `dataReceivePointByValue` or `dataReceivePointByArgument` shall refer to an `RPortPrototype` or `PRPortPrototype` that is typed by either a `SenderReceiverInterface` or an `NvDataInterface` .

⌋()

**[constr_2006] Number of `AsynchronousServerCallResultPoint` referencing to one `AsynchronousServerCallPoint`** ⌈ The `AsynchronousServerCallPoint` may be be referenced by at most one `AsynchronousServerCallResultPoint` .

If the reference exists this means that only the `RunnableEntity` with this `AsynchronousServerCallResultPoint` can fetch the result of the asynchronous server invocation of this particular `AsynchronousServerCallPoint` .

⌋()

**[constr_2007] Consistency of `typeDefinition` attribute** ⌈ All `PerInstanceMemory` s of the same `SwcInternalBehavior` with identical `type` attribute shall define an identical `typeDefinition` attribute as well.

⌋()

**[constr_2009] Supported kinds of `PortPrototype` s of a `NvBlockSwComponentType`** ⌈ With respect to external communication, `NvBlockSwComponentType` is limited to the definition of the following kinds of `PortPrototype` :

- `PortPrototype` s typed by either `NvDataInterface` s or `ClientServerInterface` s
- `RPortPrototype` s typed by `ModeSwitchInterface` s

⌋()

**[constr_2010] Connections between `SwComponentPrototype` s of type `NvBlockSwComponentType`** ⌈ The existence of `SwConnector` s that refer to `Port-`

Prototype s belonging to SwComponentPrototype s where both are typed by NvBlockSwComponentType is not permitted.

⌋()

**[constr_2011] Connections between SwComponentPrototype s typed by NvBlockSwComponentType and SwComponentPrototype s typed by other AtomicSwComponentType s** ⌈ The *nv data* PortPrototype s of the SwComponentPrototype typed by an NvBlockSwComponentType are either connected with PortPrototype s typed by NvDataInterface s or SenderReceiverInterface s of other AtomicSwComponentType .

⌋()

**[constr_2012] Compatibility of ImplementationDataType s used for ramBlock and romBlock** ⌈ The ramBlock and the romBlock shall have compatible ImplementationDataType s to ensure, that the NVRAM Block default values in the ROM Block can be copied into the RAM Block .

⌋()

**[constr_2013] Compatibility of ImplementationDataType s for NvBlockDataMapping** ⌈ The NvBlockDataMapping is only valid if the ImplementationDataType of the referenced VariableDataPrototype or ImplementationDataTypeElement in the role nvRamBlockElement is compatible to the ImplementationDataType used to type the VariableDataPrototype aggregated by NvBlockDataMapping in the role writtenNvData , writtenReadNvData , or readNvData .

⌋()

**[constr_2014] Limitation of RoleBasedPortAssignment . role in NvBlockDescriptor s** ⌈ The role has to be set to a valid name of the *Standardized AUTOSAR Interface* used for the *NVRAM Manager* e.g. *NvMNotifyJobFinished* or *NvMNotifyInit Block* .

⌋()

**[constr_2015] Limitation of SwcInternalBehavior of a NvBlockSwComponentType** ⌈ The SwcInternalBehavior of a NvBlockSwComponentType is only permitted to define

- OperationInvokedEvent s
- RunnableEntity s triggered by OperationInvokedEvent s (server RunnableEntity s)
- RunnableEntity s which defines only the mandatory attributes symbol and canBeInvokedConcurrently
- PortAPIOption s defining PortDefinedArgumentValue s

- TimingEvent s (which may include references to ModeDeclaration s in the role disabledMode )
- DataReceivedEvent s (which may include references to ModeDeclaration s in the role disabledMode )
- SwcModeSwitchEvent s
- RunnableEntity s triggered by TimingEvent s
- RunnableEntity s triggered by DataReceivedEvent s
- RunnableEntity s triggered by SwcModeSwitchEvent s
- DataTypeMappingSet

⌋()

**[constr_2016] Connections between SwComponentPrototype s of type ServiceProxySwComponentType** ⌈ A connection between PortPrototype s belonging to SwComponentPrototype s where both are typed by ServiceProxySwComponentType is not permitted.

⌋()

**[constr_2017] Ports of ServiceProxySwComponentType s** ⌈ ServiceProxySwComponentType is only permitted to define

- RPortPrototype s that are typed by SenderReceiverInterface or
- PortPrototype s that are typed by a PortInterface where the isService attribute is set to true.

⌋()

**[constr_2018] Supported remote communication of a ServiceProxySwComponentType** ⌈ For remote communication, ServiceProxySwComponentType can have only RPortPrototype s typed by SenderReceiverInterface s in a 1:n communication scenario.

⌋()

**[constr_2019] ServiceSwComponentType shall have service ports only** ⌈ In the case of ServiceSwComponentType , all aggregated PortPrototype s need to have an isOfType relationship to a PortInterface which has its isService attribute set to true . The exceptions described in TPS_SWCT_01572 , TPS_SWCT_01579 and TPS_SWCT_01580 apply.

⌋()

**[constr_2020] dataReadAccess can not be used for queued communication** ⌈ The swImplPolicy of the VariableDataPrototype referenced by a VariableAccess in role dataReadAccess shall **not** be set to queued .

⌋()

**[constr_2021] `WaitPoint` referencing a `DataReceivedEvent` can not be used for non-queued communication** ⌈ A `WaitPoint` referencing a `DataReceivedEvent` is permitted **if and only if** the `swImplPolicy` of the `VariableDataPrototype` referenced by this `DataReceivedEvent` is set to `queued` .

⌋()

**[constr_2022] Mutually exclusive use of `SynchronousServerCallPoint` s and `AsynchronousServerCallPoint` s** ⌈ A `ClientServerOperation` of a particular `RPortPrototype` shall be mutually exclusive referenced by either a `SynchronousServerCallPoint` s or an `AsynchronousServerCallPoint` s.

⌋()

**[constr_2023] Consistency of `timeout` values** ⌈ The `timeout` values of all `ServerCallPoint` s referencing the same instance of `ClientServerOperation` in a `RPortPrototype` shall be identical.

⌋()

**[constr_2024] `enableTakeAddress` is restricted to single instantiation** ⌈ The definition of a `PortAPIOption` with `enableTakeAddress` set to `true` is only permitted for software-components where the attribute `SwcInternalBehavior` . `supportsMultipleInstantiation` is set to `false` .

⌋()

**[constr_2026] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `writtenLocalVariable` and `readLocalVariable`** ⌈ A `VariableDataPrototype` in the `localVariable` reference needs to be owned by the same `SwcInternalBehavior` as this `RunnableEntity` belongs to, and the referenced `VariableDataPrototype` has to be defined in the role `implicitInterRunnableVariable` or `explicitInterRunnableVariable` .

⌋()

**[constr_2027] `SwcServiceDependency` shall be defined for service ports only** ⌈ A `PortPrototype` that is referenced by a `SwcServiceDependency` via `assignedPort` or via `assignedData` shall be typed by a `PortInterface` that has `isService` set to `true` .

This rule does **not** apply to `PortPrototype` s referenced by a `RoleBasedPortAssignment` where the attribute `role` is set to any of the following values:

- `NvMService`
- `NvMNotifyJobFinished`
- `NvMNotifyInitBlock`
- `NvMAdmin`
- `NvMMirror`

- NvDataPort

Furthermore, the rule does **not** apply to the case described in TPS_SWCT_01579 , TPS_SWCT_01580 , as well as TPS_SWCT_01572 .

⌋()

**[constr_2028] staticMemory is restricted to single instantiation** ⌈ The staticMemory is only supported if the attribute supportsMultipleInstantiation of the owning SwcInternalBehavior is set to false

⌋()

**[constr_2029] shortName of constantMemory and staticMemory** ⌈ The shortName of a VariableDataPrototype in role staticMemory or a ParameterDataPrototype in role constantMemory has to be equal with the 'C' identifier of the described variable resp. constant.

⌋()

**[constr_2030] AsynchronousServerCallResultPoint combined with WaitPoint shall belong to the same RunnableEntity** ⌈ A WaitPoint referencing a AsynchronousServerCallReturnsEvent as well as a AsynchronousServerCallResultPoint referenced by said AsynchronousServerCallReturnsEvent shall be aggregated by the same RunnableEntity .

⌋()

**[constr_2031] Period of TimingEvent shall be greater than 0** ⌈ The value of the attribute period of TimingEvent shall be greater than 0.

⌋()

**[constr_2033] Timeout of DataSendCompletedEvent** ⌈ The timeout value of a WaitPoint associated with a DataSendCompletedEvent shall have the same value as the corresponding value of TransmissionAcknowledgementRequest . timeout .

⌋()

**[constr_2034] SwAddrMethod referenced by RunnableEntity s or BswSchedulableEntity s** ⌈ RunnableEntity s and BswSchedulableEntity s shall not reference a SwAddrMethod which attribute memoryAllocationKeywordPolicy is set to addrMethodShortNameAndAlignment .

⌋()

**[constr_2035] swImplPolicy for VariableDataPrototype in SenderReceiverInterface** ⌈ The overriding swImplPolicy attribute value of a VariableDataPrototype in SenderReceiverInterface shall be standard , queued or measurementPoint .

⌋()

**[constr_2036] swImplPolicy for VariableDataPrototype in NvDataInterface** ⌈ The overriding swImplPolicy attribute value of a VariableDataPrototype in NvDataInterface shall be standard .

⌋()

**[constr_2037] swImplPolicy for VariableDataPrototype in the role ramBlock** ⌈ The overriding swImplPolicy attribute value of a VariableDataPrototype in the role ramBlock shall be standard .

⌋()

**[constr_2038] swImplPolicy for VariableDataPrototype in the role implicitInterRunnableVariable** ⌈ The overriding swImplPolicy attribute value of a VariableDataPrototype in the role implicitInterRunnableVariable shall be standard .

⌋()

**[constr_2039] swImplPolicy for VariableDataPrototype in the role explicitInterRunnableVariable** ⌈ The overriding swImplPolicy attribute value of a VariableDataPrototype in the role explicitInterRunnableVariable shall be standard .

⌋()

**[constr_2040] swImplPolicy for VariableDataPrototype in the role arTypedPerInstanceMemory** ⌈ The overriding swImplPolicy attribute value of a VariableDataPrototype in the role arTypedPerInstanceMemory shall be standard or measurementPoint .

⌋()

**[constr_2041] swImplPolicy for VariableDataPrototype in the role staticMemory** ⌈ The overriding swImplPolicy attribute value of a VariableDataPrototype in the role staticMemory shall be standard or measurementPoint .

⌋()

**[constr_2042] swImplPolicy for ParameterDataPrototype in ParameterInterface** ⌈ The overriding swImplPolicy attribute value of a ParameterDataPrototype in ParameterInterface shall be standard , const or fixed .

⌋()

**[constr_2043] swImplPolicy for ParameterDataPrototype in the role romBlock** ⌈ The overriding swImplPolicy attribute value of a ParameterDataPrototype in the role romBlock shall be standard .

⌋()

**[constr_2044] `swImplPolicy` for `ParameterDataPrototype` in the role `sharedParameter`** ⌈ The overriding `swImplPolicy` attribute value of a `ParameterDataPrototype` in the role `sharedParameter` shall be `standard` , `const` .

⌋()

**[constr_2045] `swImplPolicy` for `ParameterDataPrototype` in the role `perInstanceParameter`** ⌈ The overriding `swImplPolicy` attribute value of a `ParameterDataPrototype` in the role `perInstanceParameter` shall be `standard` , `const` .

⌋()

**[constr_2046] `swImplPolicy` for `ParameterDataPrototype` in the role `constantMemory`** ⌈ The overriding `swImplPolicy` attribute value of a `ParameterDataPrototype` in the role `constantMemory` shall be `standard` , `const` or `fixed` .

⌋()

**[constr_2047] `swImplPolicy` for `ArgumentDataPrototype`** ⌈ The overriding `swImplPolicy` attribute value of a `ArgumentDataPrototype` shall be `standard` .

⌋()

**[constr_2048] `swImplPolicy` for `SwServiceArg`** ⌈ The overriding `swImplPolicy` attribute value of a `SwServiceArg` shall be `standard` or `const` .

⌋()

**[constr_2049] Different `ModeDeclarationGroup` s shall have different `shortName` s.** ⌈ A software component is not allowed to type multiple `PortPrototype` s with `ModeSwitchInterface` s where the contained `ModeDeclarationGroupPrototype` s are referencing `ModeDeclarationGroup` s with identical `shortName` s but different `ModeDeclaration` s.

⌋()

**[constr_2050] Mandatory information of a `SwAxisCont`** ⌈ If the attribute `swAxisCont` is defined for an `ApplicationValueSpecification` the `SwAxisCont` shall define one `swAxisIndex` value and one `swArraysize` value per dimension, even in the case when the owning `ApplicationValueSpecification` defines only the content of a single dimensional object like a CURVE.

⌋()

**[constr_2051] Mandatory information of a `SwValueCont`** ⌈ If the attribute `swValueCont` is defined for an `ApplicationValueSpecification` the `SwValueCont` shall always define the attribute `swArraysize` if the `ApplicationValueSpecification` is of `category` `CURVE` , `MAP` , `CUBOID` , `CUBE_4` , `CUBE_5` , `COM_AXIS` , `RES_AXIS` , or `VAL_BLK` .

⌋()

**[constr_2052] Values of `swArraysize` and the number of values provided by `swValuesPhys` shall be consistent.** ⌈ `swValuesPhys` shall define as many numbers of values as the `swArraysize` defines.

In other words, in the bound model the number of descendants ( `v` , or `vf` , or `vt` , or `vtf` ) shall be identical to the number of elements of the related `DataPrototype` typed by an `ApplicationPrimitiveDataType` .

If several `swArraysize` values are provided these have to be multiplied in order to get the total number of `swValuesPhys` values.

⌋()

**[constr_2053] Consistency between `role IUMPRNumerator` and `ObdRatioServiceNeeds . connectionType`** ⌈ If a `SwcServiceDependency` with a `ObdRatioServiceNeeds` is defined and the attribute `connectionType` of the contained `ObdRatioServiceNeeds` is set to `ObdRatioConnectionKindEnum . apiUse` a `RoleBasedPortAssignment` with the `role` value `IUMPRNumerator` shall be defined.

If the attribute `connectionType` of the contained `ObdRatioServiceNeeds` is set to `ObdRatioConnectionKindEnum . observer` the `role` value `IUMPRNumerator` is not applicable.

⌋()

**[constr_2054] Valid targets of `rptSystem`** ⌈ The `System` referenced in the role `rptSystem` shall be of `category` `RPT_SYSTEM` .

⌋()

**[constr_2055] Valid targets of `byPassPoint` and `rptHook` reference** ⌈ Depending on the `category` value the targets of `byPassPoint` and `rptHook` references are restricted according table `table_3a_Category_of_RptContainers` .

⌋()

**[constr_2056] Consistency of `RapidPrototypingScenario` with respect to `rptSystem` and `rptArHook` references** ⌈ Within one `RapidPrototypingScenario` all `rptSystem` references shall point to instances in one and only one `System` and if existent all `rptArHook` shall point to instances in one other and only one other `System` .

⌋()

**[constr_2057] Mandatory information of a `RuleBasedAxisCont`** ⌈ If the attribute `swAxisCont` is defined for an `ApplicationRuleBasedValueSpecification` the `RuleBasedAxisCont` shall define one `swAxisIndex` value and one `swArraysize` value per dimension, even in the case when the owning `ApplicationRuleBasedValueSpecification` defines only the content of a single dimensional object like a `CURVE` .

⌋()

**[constr_2058] Mandatory information of a `RuleBasedValueCont`** ⌈ If the attribute `swValueCont` is defined for an `ApplicationRuleBasedValueSpecification` the `RuleBasedValueCont` shall define always the attribute `swArraysize` if the `ApplicationRuleBasedValueSpecification` is of `category` `CURVE` , `MAP` , `CUBOID` , `CUBE_4` , `CUBE_5` , `COM_AXIS` , `RES_AXIS` , `VAL_BLK` or `ARRAY` .

⌋()

**[constr_2535] Target of an `autosarParameter` in `AutosarParameterRef` shall refer to a parameter** ⌈ Except for the specifically described cases where constr_1173 applies the target of `autosarParameter` (which in fact is an instance ref) in `AutosarParameterRef` shall either be or be nested in `ParameterDataPrototype` . This means that the target shall either be a `ParameterDataPrototype` or an `ApplicationCompositeElementDataPrototype` that in turn is owned by a `ParameterDataPrototype` .

⌋()

**[constr_2536] Target of an `autosarVariable` in `AutosarVariableRef` shall refer to a variable** ⌈ The target of `autosarVariable` (which in fact is an instance ref) in `AutosarVariableRef` shall either be or be nested in `VariableDataPrototype` . This means that the target shall either be a `VariableDataPrototype` or an `ApplicationCompositeElementDataPrototype` that in turn is owned by a `VariableDataPrototype` .

⌋()

**[constr_2544] Limits need to be consistent** ⌈

- The limits of `ApplicationDataType` shall be inside of the definition range of the `CompuMethod`

  The `CompuMethod` needs to be applicable for limits of an `ApplicationDataType` . The reason is that the internal representation of the limits for the `ApplicationDataType` are calculated by applying the `CompuMethod` .

- The such defined internal limits of the `ApplicationDataType` shall be within or equal the `internalConstrs` of the mapped `ImplementationDataType` .

- The limits of the `ImplementationDataType` shall be within or equal to the limits defined by the size of the `BaseType` .

⌋()

**[constr_2545] `invalidValue` shall fit in the specified ranges** ⌈ The `invalidValue` shall be in the range of the `ImplementationDataType` .

⌋()

**[constr_2548] Data constraint of value axis shall match** ⌈ The values compliant to `SwDataDefProps . dataConstr` shall be also be compliant to `SwDataDefProps . valueAxisDataType . swDataDefProps . dataConstr .`

In other words `SwDataDefProps . dataConstr` win over but are not allowed to relax `SwDataDefProps . valueAxisDataType . swDataDefProps . dataConstr` but are not allowed

⌋()

**[constr_2549] Units of input axis shall be consistent** ⌈ The units specified in the context of an input axis shall be compatible, even if there is a precedence rule.

⌋()

**[constr_2550] Units of value axis shall be consistent** ⌈ The units specified in the context of value axis shall be the same, even if there is a precedence rule.

⌋()

**[constr_2561] Application of `DataConstrRule . constrLevel`** ⌈ `DataConstrRule . constrLevel` is limited to

**0:** This represents so called “hard limits”. They shall always be specified.

**1:** This represents so called “soft limits”. Soft limits may be violated after confirmation by the user of an MCD-System.

Other values may exist, but the semantics is outside of the AUTOSAR scope.

⌋()

**[constr_4000] Local communication of mode switches** ⌈ Ports with `ModeSwitchInterface` s cannot be connected across ECU boundaries.

⌋()

**[constr_4002] Unambiguous mapping of modes to data types** ⌈ Within one `DataTypeMappingSet` , a `ModeDeclarationGroup` shall not be mapped to different `ImplementationDataType` s.

⌋()

**[constr_4003] Semantics of `SwcModeSwitchEvent`** ⌈ If the value of `SwcModeSwitchEvent . activation` is `onTransition` then `SwcModeSwitchEvent` shall refer to two different `ModeDeclaration` s belonging to the same instance of `ModeDeclarationGroup` .

Their order defines the direction of the transition from one mode into another. In all other cases `SwcModeSwitchEvent` shall refer to exactly one `ModeDeclaration` .

⌋()

**[constr_4004] Context of `SenderReceiverAnnotation`** ⌈ A `SenderReceiverAnnotation` shall only be aggregated by a `PortPrototype` typed by a `SenderReceiverInterface` .

⌋()

**[constr_4005] Context of `ClientServerAnnotation`** ⌈ A `ClientServerAnnotation` shall only be aggregated by a `PortPrototype` typed by a `ClientServerInterface` .

⌋()

**[constr_4006] Context of `ParameterPortAnnotation`** ⌈ A `ParameterPortAnnotation` shall only be aggregated by a `PPortPrototype` owned by a `ParameterSwComponentType` .

⌋()

**[constr_4007] Context of `ModePortAnnotation`** ⌈ A `ModePortAnnotation` shall only be aggregated by a `PortPrototype` typed by a `ModeSwitchInterface` .

⌋()

**[constr_4008] Context of `TriggerPortAnnotation`** ⌈ A `TriggerPortAnnotation` shall only be aggregated by a `PortPrototype` typed by a `TriggerInterface` .

⌋()

**[constr_4009] Context of `NvDataPortAnnotation`** ⌈ An `NvDataPortAnnotation` shall only be aggregated by a `PortPrototype` typed by an `NvDataInterface` .

⌋()

**[constr_4010] Context of `DelegatedPortAnnotation`** ⌈ A `DelegatedPortAnnotation` shall only be aggregated by a `PortPrototype` aggregated by a `CompositionSwComponentType` .

⌋()

**[constr_4012] Timeout of `ModeSwitchedAckEvent`** ⌈ The timeout value of a `WaitPoint` associated with a `ModeSwitchedAckEvent` shall be equal to the corresponding `ModeSwitchedAckRequest.timeout` .

⌋()

**[constr_4035] `ValueSpecification` shall fit into data type** ⌈ An instance of `ValueSpecification` which is used to assign a value to a software object typed by an `AutosarDataType` shall fit into this `AutosarDataType` without losing information.

⌋()

**[constr_4082] `RunnableEntity . reentrancyLevel` shall not be set.** ⌈ The optional attribute `reentrancyLevel` shall not be set for a `RunnableEntity` . This attribute would define more specific reentrancy features than the mandatory attribute `canBeInvokedConcurrently` . These features are currently only supported for Basic Software.

⌋()

## 2.18 TPS_StandardizationTemplate

**[constr_2500] `PortInterface` s shall be of same kind** ⌈ Both objects ( `PortInterface` s) referenced by a blueprint mapping for port interfaces (represented by `BlueprintMapping` ) shall be of the same kind (e.g. both shall be `SenderReceiverInterface` s). In other words both interfaces shall be instances of the same meta class.

⌋()

**[constr_2526] `PortInterface` need to be compatible to the blueprints** ⌈ `PortInterface` shall be compatible to their respective blueprints according to the compatibility rules.

⌋()

**[constr_2527] Blueprints shall live in package of a proper category** ⌈ As explained in detail in the [10] , model artifacts (in this case `PortPrototypeBlueprint` and incompletely specified `PortInterface` s) created for the purpose of becoming blueprints shall reside in an `ARPackage` of category `BLUEPRINT` .

⌋()

**[constr_2528] `PortPrototype` s shall not refer to blueprints of a `PortInterface`** ⌈ A port `PortPrototype` shall not reference a `PortInterface` which lives in a package of category BLUEPRINT.

⌋()

**[constr_2529] `PortPrototypeBlueprint` s and derived `PortPrototype` s shall reference proper `PortInterface` s** ⌈ A `PortPrototypeBlueprint` may reference a blueprint of `PortInterface` . According to constr_2570 , a system description shall not contain blueprints. Therefore the reference to the `PortInterface` may need to be rewritten when a `PortPrototype` is derived from the blueprint.

In this case the `PortInterface` referenced by the derived `PortPrototype` shall be compatible to the `PortInterface` (which is a blueprint) referenced by the `PortPrototypeBlueprint` .

According to constr_2526 this can be ensured if the `PortInterface` referenced by the `PortPrototypeBlueprint` is the blueprint of the `PortInterface` referenced by the respective `PortPrototype` .

⌋()

**[constr_2540] Tagged text category** ⌈ The `category` of `TraceableText` shall be one of

**`SPECIFICATION_ITEM`** The text represents a particular item in the specification. Such an item is a requirement for the implementation of the software specification.

**`REQUIREMENT_ITEM`** The text represents a particular requirement. Such an item is applicable primarily in requirement specifications.

**`CONSTRAINT_ITEM`** The text represents a particular constraint. Such an item is applicable primarily in template specifications. It is similar to a specification item but represents issues that may be validated automatically e.g. by a tool.

**`IMPLEMENTATION_ITEM`** The text represents a short description of an implementation. It is applicable primarily within the `introduction` of a model element.

**`TEST_ITEM`** The text represents a short description of a test. Such an item is applicable primarily in test specifications.

**`SAFETY_*`** The text represents the type of safety requirements. The allowed values (*) are defined in [TPS_SAFEX_00102] in [11] .

⌋()

**[constr_2546] References in derived model elements** ⌈ Model elements derived from blueprints shall never refer to model elements that are blueprints.

⌋()

**[constr_2553] `shortName` shall follow the pattern defined in the Blueprint** ⌈ The `shortName` respectively `symbol` of the derived objects shall follow the pattern defined in `namePattern` of the blueprint according to TPS_STDT_00086

⌋()

**[constr_2554] Derived objects shall match the blueprints** ⌈ Unless specified explicitly otherwise, the attributes of the blueprint shall appear in the derived objects. As an exception `namePattern` may **not** be copied.

⌋()

**[constr_2556] No Blueprint Motivated `VariationPoint` s in AUTOSAR Descriptions** ⌈ AUTOSAR descriptions which are not blueprints shall not have `blueprintCondition` nor `blueprintValue` .

⌋()

**[constr_2563] `BswModuleDescription` blueprints should not have a `BswInternalBehavior`** ⌈ A `BswModuleDescription` blueprint should not have a `BswInternalBehavior` since this is a matter of implementation and not subject to standardization. Exceptions might exist in vendor internal applications.

⌋()

**[constr_2564] `VariationPoint` in Blueprints of `PackageableElement`** ⌈ To support standardization, constraint [constr_2537] in [10] is relaxed for blueprints. This means in particular, that all `PackageableElement` s which inherit from `AtpBlueprint` and live in a package of category BLUEPRINT may have a `VariationPoint` . In this case `vh.latestBindingTime` is considered as `blueprintDerivationTime` even if the meta model still states `systemDesignTime` for `PackageableElement` .

⌋()

**[constr_2565] Trace shall not be nested** ⌈ Due to the intended atomicity of requirements respectively specification items, `Traceable` shall not be nested.

⌋()

**[constr_2566] Blueprintmapping shall map appropriate elements** ⌈ `BlueprintMapping` shall map elements which represent a valid pair of blueprint / derived object. In most of the cases this means that `blueprint` and `derivedObject` shall refer to objects of the same meta-class.

⌋()

**[constr_2568] `SwComponentType` s shall be of same kind** ⌈ Both objects ( `SwComponentType` s) referenced by a blueprint mapping for port interfaces (represented by `BlueprintMapping` ) shall be of the same kind (e.g. both shall be `AtomicSwComponentType` s). In other words both components shall be instances of the same meta class.

⌋()

**[constr_2569] Purely Bluprint Motivated `VariationPoint` s** ⌈ `VariationPoint` s with `vh.latestBindingTime` set to `blueprintDerivationTime` shall have only `blueprintCondition` respectively `blueprintValue` .

⌋()

**[constr_2570] No Blueprints in system descriptions** ⌈ There shall be no blueprints in system descriptions. In consequence of this blueprint elements shall be referenced only from blueprints and `AtpBlueprintMapping` s. Due to `atpUriDef` , the references from `AtpBlueprintMapping` do not need to be resolved in system descriptions.

⌋()

**[constr_2571] Outgoing references from Blueprints** ⌈ Note that outgoing references from Blueprints are basically not limited. Practically, references to objects living in a package of category EXAMPLE should not occur.

⌋()

**[constr_2589] In VFB Timing Blueprint `TDEventVfbPort` shall reference `PortPrototypeBlueprint`** ⌈ In a VFB Timing Blueprint `TDEventVfbPort` shall reference `PortPrototypeBlueprint` . In other words, a VFB Timing Description Event specified in a VFB Timing Blueprint shall always reference a Port Prototype Blueprint.

⌋()

**[constr_2590] One BlueprintPolicy is allowed** ⌈ For each attribute of a blueprint, at most one `BlueprintPolicy` is allowed.

⌋()

**[constr_2591] BlueprintPolicyNotModifiable** ⌈ If `BlueprintPolicyNotModifiable` is assigned to an attribute, then during blueprinting it is not allowed to modify the value of the attribute and all it contained content.

⌋()

**[constr_2592] No BlueprintPolicy** ⌈ If no `BlueprintPolicy` is assigned to an attribute, then arbitrary modifications are allowed while deriving from the blueprint.

⌋()

**[constr_2593] Expression for identifying the attribute a BlueprintPolicy relates to** ⌈ The expression language for identifying the related attribute of a `BlueprintPolicy` is a subset version of xpath, see [12] . For navigation over the model we use the names as they are used in XML.

⌋()

**[constr_2597] `ClientServerOperationBlueprintMapping` constraints number of arguments** ⌈ The number of arguments of the `BswModuleEntry` referenced by a `bswModuleEntry` shall be identical to the number of portDefinedArgumentBlueprints of the owning `ClientServerInterfaceToBswModuleEntryBlueprintMapping` plus the number of `ArgumentDataPrototype` s aggregated in the role argument of the `clientServerOperation`

⌋()

**[constr_2598] `ClientServerOperationBlueprintMapping` constraints the types of arguments** ⌈ The arguments in the ordered lists `bswModuleEntry` and the matching arguments in the set union of the ordered lists `portDefinedArgumentBlueprint` plus `clientServerOperation` shall result in the identical C data type definitions.

⌋()

**[constr_2603] Use of "applies to" in context of the specification level** ⌈ On specification level 1 and 2 only the requirements table including the appliesTo attribute shall be used. On the specification levels 3 and 4 only the requirements table without the appliesTo attribute shall be used. Exception: Documents of the foundation which are handled on specification level 3.

⌋()

**[constr_2604] Allowed uptraces in context of "applies to" values** ⌈ Traces to documents of upper specification levels shall be conform to the values assigned to appliesTo.

⌋()

**[constr_2608] Custom extensions shall be part of the `Documentation` that is referenced by the `Baseline`** ⌈ If a `SpecElementReference` references a custom defined specification element, then this specification element shall be part of a `Documentation` that is referenced by the `Baseline` of this `Profile` .

⌋()

**[constr_2609] Single revision per AUTOSAR standard** ⌈
The `standardRevision` may only contain a single revision per AUTOSAR standard. E.g. it is allowed to combine the AUTOSAR standards "Foundation" in revision 1.0.0 with the "Classic Platform" in revision 4.3.0. However, it is not allowed to reference the revisions 4.2.2 and 4.3.0 of the "Classic Platform" in the same `Baseline` .

⌋()

**[constr_2610] No alternativeName if matching via shortName** ⌈ The `alternativeName` shall not be set if the referenced AUTOSAR Specification Element matches the rules of `Identifier` .

⌋()

**[constr_2611] Referenced AUTOSAR Specification Elements shall be part of the AUTOSAR Specification Baseline** ⌈ If the `SpecElementReference` references an AUTOSAR specification element then the `shortName` or `alternativeName` shall match the name of the AUTOSAR specification element in a specification that is part of the revision of the standard that is specified in `Baseline` .

⌋()

**[constr_2612] `shortName` of `ConcreteClassTailoring` shall match the name of an AUTOSAR specified concrete meta-class** ⌈ `shortName` of `ConcreteClassTailoring` shall match the name of an AUTOSAR specified concrete meta-class).

⌋()

**[constr_2613] `shortName` of `AbstractClassTailoring` shall match the name of an AUTOSAR specified abstract meta-class** ⌈ `shortName` of `AbstractClassTailoring` shall match the name of an AUTOSAR specified abstract meta-class).

⌋()

**[constr_2614] `PrimitiveAttributeCondition . attribute` shall reference invariant owned `PrimitiveAttributeTailoring` , only** ⌈ The following condi-

tions need to evaluate to true for `PrimitiveAttributeCondition . attribute` :

- The referenced `PrimitiveAttributeTailoring` is owned by an `ClassContentConditional` that has no `condition` (invariant class content) **AND**
- The `ClassContentConditional` that owns the referenced `PrimitiveAttributeTailoring` and the `ClassContentConditional` that owns this `PrimitiveAttributeCondition` are owned by the same `ClassTailoring` .

⌋()

**[constr_2615] `AggregationCondition . aggregation` shall reference invariant owned `AggregationTailoring` , only** ⌈ The following conditions need to evaluate to true for `AggregationCondition . aggregation` :

- The referenced `AggregationTailoring` is owned by an `ClassContentConditional` that has no `condition` (invariant class content) **AND**
- The `ClassContentConditional` that owns the referenced `AggregationTailoring` and the `ClassContentConditional` that owns this `AggregationCondition` are owned by the same `ClassTailoring` .

⌋()

**[constr_2616] `ReferenceCondition . reference` shall reference invariant owned `ReferenceTailoring` , only** ⌈ The following conditions need to evaluate to true for `ReferenceCondition . reference` :

- The referenced `ReferenceTailoring` is owned by an `ClassContentConditional` that has no `condition` (invariant class content) **AND**
- The `ClassContentConditional` that owns the referenced `ReferenceTailoring` and the `ClassContentConditional` that owns this `ReferenceCondition` are owned by the same `ClassTailoring` .

⌋()

**[constr_2617] `ClassTailoring . variationRestriction` only applicable for «atpVariation» classes** ⌈ If the tailored meta class is not marked with stereotype «atpVariation» then `ClassTailoring . variationRestriction` shall not be defined.

⌋()

**[constr_2618] ShortName of AttributeTailoring shall match owned or inherited attributes** ⌈ The `shortName` shall match the name of an attribute that is owned or inherited by the AUTOSAR meta-class which is identified by the `ClassTailoring` that owns this `AttributeTailoring` .

⌋()

**[constr_2619] No `AttributeTailoring` for Derived or Abstract Attributes** ⌈ No `AttributeTailoring` s are allowed for `Attributes` that are marked with stereotypes «`atpDerived`» or «`atpAbstract`» .

⌋()

**[constr_2620] `shortName` of `PrimitiveAttributeTailoring` shall be a primitive attribute in the referenced Baseline** ⌈ The `shortName` of `PrimitiveAttributeTailoring` shall match the name of an AUTOSAR specified primitive attribute of the Meta-Class in the referenced Baseline.

⌋()

**[constr_2621] The `shortName` of `AggregationTailoring` shall match the name of an AUTOSAR specified aggregation of the meta-class** ⌈ The `shortName` of `AggregationTailoring` shall match the name of an AUTOSAR specified aggregation of the meta-class).

⌋()

**[constr_2622] The `shortName` of `ReferenceTailoring` shall match the name of an AUTOSAR specified reference of the meta-class** ⌈ The `shortName` of `ReferenceTailoring` shall match the name of an AUTOSAR specified reference of the meta-class).

⌋()

**[constr_2623] Referenced `SdgClass` shall be part of a `SdgDef` that is referenced by the `Baseline`** ⌈ Referenced `SdgClass` shall be part of a `SdgDef` that is referenced by the `Baseline` of this `Profile of Data Exchange Point` .

⌋()

**[constr_2624] `AttributeTailoring.variationRestriction` only applicable for «atpVariation» attributes** ⌈ If the tailored attribute is not marked with stereotype «atpVariation» then `AttributeTailoring.variationRestriction` shall not be defined.

⌋()

## 2.19 TPS_SystemTemplate

**[constr_1002] End-to-end protection does not support n:1 communication** ⌈ As the n:1 communication scenario implies that probably not all senders use the same `dataId` this scenario is explicitly not supported.

⌋()

**[constr_1198] `TriggerToSignalMapping.systemSignal` s eligible for a `TriggerToSignalMapping`** ⌈ In the context of a `TriggerToSignalMapping` , it is

only possible to refer to a TriggerToSignalMapping . systemSignal that in turn is referenced by an ISignal with attribute length set to 0.

⌋()

**[constr_1199] ISignal s relating to systemSignal s eligible for a TriggerToSignalMapping** ⌈ An ISignal used to reference a systemSignal that in turn is referenced by a TriggerToSignalMapping shall also be referenced by an ISignalToIPduMapping where the attribute updateIndicationBitPosition is defined.

⌋()

**[constr_1207] Existence of the attribute DataMapping . communicationDirection in the context of a SenderReceiverInterface or TriggerInterface** ⌈ The following condition shall be fulfilled regarding the existence and values of the attribute DataMapping . communicationDirection that refers to a PortPrototype typed by a SenderReceiverInterface or TriggerInterface as the context PortPrototype :

- If the DataMapping refers to a PRPortPrototype as the context PortPrototype the attribute DataMapping . communicationDirection shall exist.
- If the DataMapping refers to a PPortPrototype as the context PortPrototype the attribute DataMapping . communicationDirection may exist. If the attribute exists its value shall be set to out .
- If the DataMapping refers to an RPortPrototype as the context PortPrototype the attribute DataMapping . communicationDirection may exist. If the attribute exists its value shall be set to in .

⌋()

**[constr_1265] DoIpGidSynchronizationNeeds can only exist once per ECU_EXTRACT** ⌈ Within the context of one System of category ECU_EXTRACT , there can only be at most one DoIpGidSynchronizationNeeds .

⌋()

**[constr_1266] DoIpGidNeeds can only exist once per ECU_EXTRACT** ⌈ Within the context of one System of category ECU_EXTRACT , there can only be at most one DoIpGidNeeds .

⌋()

**[constr_1267] DoIpActivationLineNeeds can only exist once per ECU_EXTRACT** ⌈ Within the context of one System of category ECU_EXTRACT , there can only be at most one DoIpActivationLineNeeds .

⌋()

**[constr_1367] periodicResponseUudt . periodicResponseUudt shall only refer to a DcmIPdu** ⌈ If the role periodicResponseUudt exists then every

PduTriggering referenced in the role periodicResponseUudt shall only refer to a DcmIPdu .

⌋()

**[constr_1368] Limitation of the target of references from DiagnosticConnection** ⌈ DiagnosticConnection shall only reference (via the indirection created by TpConnectionIdent ) the following sub-classes of the meta-class TpConnection :

- CanTpConnection
- FlexrayTpConnection
- FlexrayArTpConnection
- DoIpTpConnection

⌋()

**[constr_1369] CommunicationConnector s shall be attached to the same CommunicationCluster** ⌈ All CommunicationConnector s referenced from GlobalTimeMaster and GlobalTimeSlave s aggregated in one GlobalTimeDomain shall be referenced in the role commConnector by the same PhysicalChannel aggregated by the same CommunicationCluster .

⌋()

**[constr_1370] Consistency of GlobalTimeDomain** ⌈ The GlobalTimeSlave referenced in the role GlobalTimeGateway . slave and the GlobalTimeMaster referenced in the role GlobalTimeGateway . master shall **not** be aggregated by the same GlobalTimeDomain .

⌋()

**[constr_1371] Consistency of attribute host** ⌈ Within the context of an aggregating GlobalTimeDomain , the CommunicationConnector s referenced in the role GlobalTimeGateway . master . communicationConnector and GlobalTimeGateway . slave . communicationConnector shall be aggregated by the same EcuInstance that is referenced in the role GlobalTimeGateway . host .

⌋()

**[constr_1372] Consistency of attribute globalTimePduTriggering** ⌈ Within the context of an aggregating GlobalTimeDomain , the globalTimePduTriggering shall be owned by PhysicalChannel that is also referencing the CommunicationConnector s referenced in the roles GlobalTimeSlave . communicationConnector and GlobalTimeMaster . communicationConnector .

⌋()

**[constr_1373] GlobalTimeMaster with attribute isSystemWideGlobalTimeMaster set to TRUE** ⌈ GlobalTimeMaster with attribute isSystemWide-

GlobalTimeMaster set to TRUE shall not be referenced in the role GlobalTimeGateway.master.

⌋()

**[constr_1374] Only fan-out possible for GlobalTimeGateway** ⌈ For all GlobalTimeGateway s that refer to the same EcuInstance the condition applies that no two GlobalTimeGateway s shall refer to the same GlobalTimeMaster .

⌋()

**[constr_1387] Transmission of Variable-Size Array Data Type s by means of a Transformer** ⌈ If a Transformer is used for the transmission of a Variable-Size Array Data Type s then the Variable-Size Array Data Type shall be a "new-world" variable-size array data type according to [TPS_SWCT_01644] and [TPS_SWCT_01645] . "Old-world" dynamic-size array data types according to [TPS_SWCT_01642] and [TPS_SWCT_01643] are not supported.

⌋()

**[constr_1441] In AUTOSAR, the transmission of union data types over the network is only supported by the SOME/IP Transformer** ⌈ If an ImplementationDataType according to [TPS_SWCT_01700] , i.e. of category STRUCT that encloses an ImplementationDataTypeElement of category UNION , is used to directly or (via a DataTypeMap ) indirectly type an AutosarDataPrototype and the latter is mapped to a SystemSignal then the ISignal that references that SystemSignal shall aggregate transformationISignalProps .

⌋()

**[constr_1463] Applicable values for J1939Cluster.networkId** ⌈ The values of the attribute J1939Cluster.networkId shall always be within the interval 1..4.

⌋()

**[constr_2025] Uniqueness of symbol attributes** ⌈ With the exception of Runnable Entities that are subject to 1234 (RunnableEntities owned by NvBlockSwComponent Types), in the context of a single EcuInstance the values of the RunnableEntity.symbol in combination with the attribute symbol of the meta-class SymbolProps owned by AtomicSwComponentType of all deployed RunnableEntities shall be unique such that no two (or more) combinations of RunnableEntity.symbol and the symbol of the meta-class SymbolProps owned by AtomicSwComponentType in the role symbolProps share the same value.

⌋()

**[constr_3000] valid SenderRecCompositeTypeMapping s** ⌈ SenderReceiverToSignalGroupMapping.signalGroup.systemSignal shall point to each SystemSignal being mapped within the context of SenderReceiverToSignalGroupMapping .

In other words: For each SystemSignal referenced in the role SenderReceiverToSignalGroupMapping . signalGroup . systemSignal there shall be either a reference in the role SenderRecRecordElementMapping . systemSignal or a reference in the role SenderRecArrayElementMapping . systemSignal aggregated by the same SenderReceiverToSignalGroupMapping that refers to this SystemSignal .

⌋()

**[constr_3002] valid swcToImplMapping** ⌈ The referenced SwcImplementation refers to a SwcInternalBehavior that is part of a AtomicSwComponentType . The same AtomicSwComponentType shall be the type of the referenced SwComponentPrototype .

SwcToImplMapping.componentImplementation.behavior.component == SwcToImplMapping.component.type

⌋()

**[constr_3003] Number of CAN channels** ⌈ CAN clusters shall aggregate exactly one PhysicalChannel .

⌋()

**[constr_3004] Clustering and separation must be exclusive** ⌈ Clustering and separation must be exclusive, i.e. it SHALL NOT be possible that two SwComponentPrototype s A and B are associated by a ComponentClustering and by a ComponentSeparation .

⌋()

**[constr_3005] valid EcuResourceEstimation** ⌈ The same EcuInstance shall be referenced directly from the EcuResourceEstimation and from the SwcToEcuMapping :

EcuResourceEstimation.swCompToEcuMapping.ecuInstance == EcuResourceEstimation.ecuInstance

⌋()

**[constr_3006] valid EcuMapping** ⌈ The referenced hwCommunicationController and hwCommunicationPort shall be part of the referenced ecu .

ECUMapping . ecu .nestedElement contains ECUMapping . commControllerMapping . hwCommunicationController

ECUMapping . ecu .nestedElement contains ECUMapping . hwPortMapping . hwCommunicationPort

⌋()

**[constr_3007] `selectorFieldCode` s for dynamic part alternatives** ⌈ The `selectorFieldCode` s for the dynamic part alternatives within one `MultiplexedIPdu` shall differ from each other.

⌋()

**[constr_3008] `EcuInstance` subelements** ⌈ The `CommunicationConnector` and the `CommunicationController` that is referenced by the `CommunicationConnector` must be owned by the same `EcuInstance` .

⌋()

**[constr_3009] Overlapping of `ISignal` s is prohibited** ⌈ `ISignal` s mapped to an `ISignalIPdu` shall not overlap.

⌋()

**[constr_3010] `ISignalIPdu` length shall not be exceeded** ⌈ The combined length of all `ISignal` s and `updateIndicationBitPosition` s that are mapped into an `ISignalIPdu` shall not exceed the defined `Pdu length` .

⌋()

**[constr_3011] Overlapping of `updateIndicationBits` of `ISignal` s is prohibited** ⌈ The `updateIndicationBitPosition` for an `ISignal` in an `ISignalIPdu` shall not overlap with other `updateIndicationBitPosition` s or `ISignal` locations.

⌋()

**[constr_3012] Overlapping of `Pdu` s is prohibited** ⌈ `Pdu` s mapped to a `Frame` shall NOT overlap.

⌋()

**[constr_3013] Frame length shall not be exceeded** ⌈ The combined length of all `Pdu` s that are mapped into a `Frame` shall not exceed the defined `Frame` length.

⌋()

**[constr_3014] Overlapping of updateIndicationBits for `Pdu` s is prohibited** ⌈ The `updateIndicationBitPosition` for a `Pdu` in a `Frame` shall NOT overlap with other `updateIndicationBitPosition` s and `Pdu` locations.

⌋()

**[constr_3015] Number of LIN channels** ⌈ LIN clusters shall aggregate exactly one `LinPhysicalChannel` .

⌋()

**[constr_3018] Number of FlexRay channels** ⌈ A `FlexrayCluster` shall use either one `FlexrayPhysicalChannel` with `channelName` set to either `channelA` or

channelB or else two FlexrayPhysicalChannel s with one channelName channelA and one channelName channelB .

⌋()

**[constr_3019] In the flat ECU extract each required interface must be satisfied by connected provided interfaces** ⌈ In case of the flat System with category ECU_EXTRACT all VariableDataPrototype s specified by the SenderReceiverInterface of the RPortPrototype need to be supplied by some of the PPortPrototype s being connected with SwConnector s.

⌋()

**[constr_3020] communicationDirection of containedISignalIPduGroup s** ⌈ The value of the attribute communicationDirection of containedISignalIPduGroup must be identical to the value of the attribute communicationDirection of the enclosing ISignalIPduGroup .

⌋()

**[constr_3021] Mapping of SensorActuatorSwComponent s to SensorActuator HwElement s** ⌈ Only SwComponentPrototype s that are typed by SensorActuatorSwComponentType shall be mapped to a HwElement with category SensorActuator via the controlledHwElement relation.

⌋()

**[constr_3024] Usage of triggeredWithoutRepetition and triggeredOnChangeWithoutRepetition is not allowed for signal groups and group signals.** ⌈ The values triggeredWithoutRepetition and triggeredOnChangeWithoutRepetition shall not be used if the ISignalToIPduMapping refers to an ISignalGroup or an ISignal which is part of an ISignalGroup (group signal).

⌋()

**[constr_3025] Usage of NPdu s in TpConnection s** ⌈ In case several TpConnection s use the same Frame ID for their communication needs only one NPdu element per Frame Id shall exist. This constraint applies for all supported AUTOSAR transport protocols (CanTp, LinTp, FrTp, FrArTp and J1939Tp).

⌋()

**[constr_3027] Existence of ecuExtractVersion** ⌈ In case the category of the System is SYSTEM_EXTRACT or ECU_EXTRACT the ecuExtractVersion attribute shall be defined.

⌋()

**[constr_3028] FibexElements** ⌈ Each FibexElement that is used in the System Description shall be referenced by the System element in the role FibexElement .

⌋()

**[constr_3029] Assign-Frame command usage** ⌈ For the LIN 2.0 Assign-Frame command the `LinConfigurableFrame` list shall be used. For the LIN 2.1 Assign-Frame-PID-Range command the `LinOrderedConfigurableFrame` list shall be used.

⌋()

**[constr_3030] valid relationship between `ECUMapping` and `EcuInstance`** ⌈ If an `EcuInstance` is assigned to a `HwElement` the `EcuInstance` shall belong to the same `System` as the `ECUMapping` .

⌋()

**[constr_3031] Complete System Description does not have ports on the outermost composition** ⌈ In a complete `System` with `category ABSTRACT_SYSTEM_DESCRIPTION` or `System` with `category SYSTEM_DESCRIPTION` this outermost `CompositionSwComponentType` has the unique feature that it doesn't have any outside ports, but all the SWC contained in it are connected to each other and fully specified by their `SwComponentType` s, `PortPrototype` s, `PortInterface` s, `VariableDataPrototype` s, `InternalBehavior` etc.

⌋()

**[constr_3034] Values of `LinSlaveConfig` and `LinSlave` attributes** ⌈ The values of attributes of `LinSlaveConfig` and `LinSlave` shall be identical for each `LinSlaveConfig` that points to a `LinSlave` .

⌋()

**[constr_3035] CanNm user data configuration in case NID/CBV are enabled** ⌈ If NID/CBV are enabled ( `nmCbvPosition` and `nmNidPosition` are configured), there shall not be any user data configured at the position of the respective NID/CBV bytes.

⌋()

**[constr_3036] `Pdu` s in CAN and LIN Frames** ⌈ CAN Frames and LIN Frames shall only contain one `Pdu` .

⌋()

**[constr_3037] maximum `Frame frameLength` for CAN and LIN** ⌈ For CAN and LIN the maximum `frameLength` is 8 bytes and 64 bytes in case of CAN FD.

⌋()

**[constr_3038] maximum `Frame frameLength` for FlexRay** ⌈ For FlexRay the maximum `frameLength` is 254 bytes.

⌋()

**[constr_3039] `pncIdentifier` range** ⌈ The `pncIdentifier` value shall be in the range of 8..63.

⌋()

**[constr_3040] Restriction of `pncIdentifier` values** ⌈ The `pncIdentifier` value shall be within the range described by `pncVectorOffset` and `pncVectorLength` .

⌋()

**[constr_3041] `pncVectorOffset` range** ⌈ The `pncVectorOffset` value shall be in the range of 1..7.

⌋()

**[constr_3042] `pncVectorLength` range** ⌈ The `pncVectorLength` value shall be in the range of 1..7.

⌋()

**[constr_3043] `pncVector` configuration in AUTOSAR Com** ⌈ The `pncVector` shall be configured as UINT8_N signal in AUTOSAR Com.

⌋()

**[constr_3044] CBV configuration in case partial network is used** ⌈ In case a partial network is used the control bit vector (CBV) shall be defined in Byte 0 of the `NmPdu` ( `nmCbvPosition` = 0).

⌋()

**[constr_3045] Signal content evaluation vs. Mode evaluation** ⌈ The mode evaluation and the signal content evaluation shall not be used in the same `IPdu` . A mix of these two types is not allowed.

⌋()

**[constr_3046] Consistency of `TransmissionModeCondition.iSignalInIPdu`** ⌈ The `ISignalToIPduMapping` referenced by the `TransmissionModeCondition` in the role `iSignalInIPdu` shall belong to the same `ISignalIPdu` as the `TransmissionModeCondition` .

⌋()

**[constr_3047] Uniqueness of `macMulticastAddress` es** ⌈ A `macMulticastAddress` shall be unique in a particular `EthernetCluster` .

⌋()

**[constr_3048] Range of `vlanIdentifier`** ⌈ The allowed values of `vlanIdentifier` range from 0 to 4095.

⌋()

**[constr_3049] Role of `SystemSignal` in inter-ECU client server communication with clients located on different ECUs in case of networks other than Ethernet** ⌈ In case of a n:1 inter-ECU client server communication with clients located on different ECUs different `SystemSignal` s shall be used for each Ecu.

⌋()

**[constr_3050] `J1939Cluster` uses exactly one `CanPhysicalChannel`** ⌈ A `J1939Cluster` shall aggregate exactly one `CanPhysicalChannel` .

⌋()

**[constr_3051] Restriction of `ISignalMapping` references** ⌈ If the `sourceSignal` references an `ISignal` then the `targetSignal` shall also reference an `ISignal` .

⌋()

**[constr_3052] Complete `ISignalMapping` of `ISignalGroup` signals** ⌈ If an `ISignalMapping` to an `ISignal` that is a member of a `ISignalGroup` exists then (see TPS_SYST_01120 ) an `ISignalMapping` to the enclosing `ISignalGroup` shall exist as well.

⌋()

**[constr_3053] Complete `ISignalMapping` of target `ISignalGroup`** ⌈ If an `ISignalGroup` is referenced by a `targetSignal` then TPS_SYST_02162 applies for each of the contained `ISignal` of that `ISignalGroup` .

⌋()

**[constr_3055] `SystemSignalGroup` in a complete System Description** ⌈ For each `SystemSignalGroup` in a complete `System` with `category SYSTEM_DESCRIPTION` exactly one `DataMapping` shall be defined ( `PPortPrototype` or `RPortPrototype` ). Preference: `PPortPrototype`

⌋()

**[constr_3057] Maximal one `BusspecificNmEcu` per `NmEcu` and bus system is allowed to be defined** ⌈ For each `NmEcu` at most one `BusspecificNmEcu` per bus system (FlexRay/Can/Udp/J1939) is allowed to be defined.

⌋()

**[constr_3058] References from `SenderRecArrayElementMapping` and from `SenderRecRecordElementMapping` to `SystemSignal` s are not allowed within a `SenderReceiverCompositeElementToSignalMapping`** ⌈ The reference from `SenderRecArrayElementMapping` to `SystemSignal` and from `SenderRecRecordElementMapping` to `SystemSignal` shall not exist if the enclosing `SenderRecCompositeTypeMapping` is owned by a `SenderReceiverCompositeElementToSignalMapping` .

⌋()

**[constr_3059] Mandatory `DataMapping` on the receiver side for elements of a composite data type** ⌈ On the receiver side, it is required that for every `ApplicationCompositeElementDataPrototype` of an `ApplicationCompositeDataType` ( `ApplicationArrayDataType` . `element` resp. `Application-`

RecordDataType . element ) that types a dataElement in a RPortPrototype or PRPortPrototype in its receiver role a DataMapping exists.

⌋()

**[constr_3060] Usage of networkRepresentationProps and physicalProps** ⌈ Usage of networkRepresentationProps and physicalProps shall follow the restrictions given in table table_3a_SwDataDefPropsForSignals .

⌋()

**[constr_3062] The EcuInstance that is referenced from a specific CouplingElement shall be connected to the same EthernetCluster as the specific CouplingElement** ⌈ The EcuInstance referenced from a specific CouplingElement in the role ecuInstance shall be connected via the CommunicationConnector and a EthernetPhysicalChannel that refers the CommunicationConnector to the EthernetCluster referenced by the specific CouplingElement in the role communicationCluster .

⌋()

**[constr_3063] Usage of portNumber and dynamicallyAssigned with value "true" is mutually exclusive** ⌈ Usage of portNumber and dynamicallyAssigned with value "true" is mutually exclusive.

⌋()

**[constr_3064] Usage of serviceInstance , eventHandler and eventGroup references** ⌈ The serviceInstance , eventHandler and eventGroup references shall only be used to describe a service based communication over the Internet Protocol. More details are described in chapter sec_3a_EthernetCommunication .

⌋()

**[constr_3065] Mapping of queued Trigger s to SystemSignal s is prohibited** ⌈ A TriggerToSignalMapping of a Trigger with swImplPolicy set to queued is prohibited.

⌋()

**[constr_3067] initValue defined in the context of ISignal** ⌈ The definition of an initValue in the context of an ISignal can only be a primitive NumericalValueSpecification or TextValueSpecification .

⌋()

**[constr_3068] DoIpPowerModeStatusNeeds in the category ECU_EXTRACT** ⌈ If and only if DoIP (i.e. any of the subclasses of DoIpServiceNeeds are present) is used on an Ecu then the DoIpPowerModeStatusNeeds shall exist exactly once in a System of category ECU_EXTRACT .

⌋()

**[constr_3069] Allowed `CanNmCluster . nmNidPosition` values** ⌈ The value of `CanNmCluster . nmNidPosition` shall only be set to either 0 or 1.

⌋()

**[constr_3070] Allowed `CanNmCluster . nmCbvPosition` values** ⌈ The value of `CanNmCluster . nmCbvPosition` shall only be set to either 0 or 1.

⌋()

**[constr_3071] `CanNmCluster . nmCbvPosition` and `CanNmCluster . nmNidPosition` shall never have the same value** ⌈ `CanNmCluster . nmCbvPosition` and `CanNmCluster . nmNidPosition` shall never have the same value.

⌋()

**[constr_3073] `nmVoteInformation` only valid for FrNm** ⌈ The `nmVoteInformation` attribute is only valid for FrNm.

⌋()

**[constr_3074] No TransmissionAcknowledgementRequest for multiple senders** ⌈ If more than one `SenderComSpec` exist (in different `PortPrototype` s on atomic level) that refer to data elements effectively mapped to the same `SystemSignal` it is not allowed that any `SenderComSpec` aggregates `transmissionAcknowledge` .

⌋()

**[constr_3078] Allowed `UdpNmCluster . nmNidPosition` values** ⌈ The value of `UdpNmCluster . nmNidPosition` shall only be set to either 0 or 1.

⌋()

**[constr_3079] Allowed `UdpNmCluster . nmCbvPosition` values** ⌈ The value of `UdpNmCluster . nmCbvPosition` shall only be set to either 0 or 1.

⌋()

**[constr_3080] `UdpNmCluster . nmCbvPosition` and `UdpNmCluster . nmNidPosition` shall never have the same value** ⌈ `UdpNmCluster . nmCbvPosition` and `UdpNmCluster . nmNidPosition` shall never have the same value.

⌋()

**[constr_3081] Value of category in `GeneralPurposePdu`** ⌈ The attribute `category` of `GeneralPurposePdu` can have the following values:

- SD (Service Discovery)
- GLOBAL_TIME
- DoIP

⌋()

**[constr_3082] Value of category in `GeneralPurposeIPdu`** ⌈ The attribute `category` of `GeneralPurposeIPdu` can have the following values:

- XCP
- SOMEIP_SEGMENTED_IPDU
- DLT

⌋()

**[constr_3083] Exactly one `AtomicSwComponentType` on an `EcuInstance` may use `GeneralCallbackEventDataChanged` / `GeneralCallbackEventStatusChange`** ⌈ The Dem only supports exactly one `AtomicSwComponentType` using `GeneralCallbackEventDataChanged` / `GeneralCallbackEventStatusChange` on one `EcuInstance` .

⌋()

**[constr_3084] Service port in the role PowerTakeOff** ⌈ Within the context of one `EcuInstance` , there can only be one service port that uses the role PowerTakeOff in the `RoleBasedPortAssignment` . `role` .

⌋()

**[constr_3085] Service port in the role CallbackDCMRequestServices** ⌈ Within the context of one `EcuInstance` , there can only be one service port that uses the role CallbackDCMRequestServices in the `RoleBasedPortAssignment` . `role` .

⌋()

**[constr_3086] Role of `SystemSignal` in n:1 sender-receiver communication** ⌈ In case of n:1 communications each sender needs to be represented by the same `SystemSignal` .

⌋()

**[constr_3087] `DataMapping` to `PRPortPrototype`** ⌈ For inter-ECU communication between `SwComponentPrototype` s which involves `PRPortPrototype` s for each `DataPrototype` there shall be one `SystemSignal` and at most two `DataMapping` s, one for each direction.

⌋()

**[constr_3088] `SystemSignal` that is not part of a `SystemSignalGroup` in a complete System Description** ⌈ For each `SystemSignal` that is not part of a `SystemSignalGroup` in a complete `System` with `category` `SYSTEM_DESCRIPTION` exactly one `DataMapping` per `communicationDirection` shall be defined ( `PPortPrototype` , `RPortPrototype` , `PRPortPrototype` ). Preference: `AbstractProvidedPortPrototype`

⌋()

**[constr_3089] `SystemSignal` that is part of exactly one `SystemSignalGroup` and is not transmitted additionally as standalone `SystemSignal` in a complete System Description** ⌈ For each `SystemSignal` that is part of exactly one `SystemSignalGroup` and is not transmitted additionally as standalone `SystemSignal` in a complete `System` with `category` `SYSTEM_DESCRIPTION` exactly one `DataMapping` per `communicationDirection` shall be defined ( `PPortPrototype` , `RPortPrototype` , `PRPortPrototype` ). Preference: `AbstractProvidedPortPrototype`

⌋()

**[constr_3090] `TpSdu` transmission on a `PhysicalChannel`** ⌈ The `IPdu` that is referenced by a `TpConnection` in the role `tpSdu` shall be referenced by exactly one `PduTriggering` aggregated on the `PhysicalChannel` of the `TpConnection` .

⌋()

**[constr_3094] Consistent `ISignalPort . communicationDirection` for `ISignalTriggering` s of `ISignalGroup` s and contained `ISignal` s** ⌈ In case the `ISignal` s contained in an `ISignalGroup` are referenced by an `ISignalTriggering` , the `communicationDirection` of the `ISignalPort` referenced by the `ISignal` 's `ISignalTriggering` shall be identical to the `communicationDirection` of the `ISignalPort` referenced by the containing `ISignalGroup` 's `ISignalTriggering` .

⌋()

**[constr_3095] canControllerFdAttributes and canControllerFdRequirements are mutually exclusive** ⌈ The existence of `canControllerFdAttributes` and `canControllerFdRequirements` is mutually exclusive.

⌋()

**[constr_3096] Allowed values for `diagnosticMessageType`** ⌈ The allowed values of `diagnosticMessageType` range from 1..57.

⌋()

**[constr_3097] Overlapping of segments of one `MultiplexedIPdu` is not allowed** ⌈ The segments defined by the `SegmentPosition` elements of one and the same `MultiplexedIPdu` - aggregated via `StaticPart` and `DynamicPart` - shall not overlap.

⌋()

**[constr_3098] Defined segments of one `MultiplexedIPdu` shall not exceed the length of the `MultiplexedIPdu`** ⌈ The segments defined by the `SegmentPosition` elements of one and the same `MultiplexedIPdu` - aggregated via `StaticPart` and `DynamicPart` - shall not exceed the length of the `MultiplexedIPdu` .

⌋()

**[constr_3099] Defined segments in a `DynamicPart` shall not exceed the length of any `DynamicPartAlternative . iPdu`** ⌈ The segments defined by the `SegmentPosition` elements aggregated in the `DynamicPart` of a `MultiplexedIPdu` shall not exceed the length of any `DynamicPartAlternative . iPdu` .

⌋()

**[constr_3100] Defined segments in a `StaticPart` shall not exceed the length of the `StaticPart . iPdu`** ⌈ The segments defined by the `SegmentPosition` elements aggregated in the `StaticPart` of a `MultiplexedIPdu` shall not exceed the length of the `StaticPart . iPdu`

⌋()

**[constr_3101] Signal representation of selector field for `DynamicPartAlternative`** ⌈ Every `ISignalIPdu` that is referenced by the `DynamicPartAlternative` shall contain an `ISignal` that represents the selector field. The selector field signal shall be located at the position that is described by the `selectorFieldLength` and `selectorFieldStartPosition` .

⌋()

**[constr_3102] Restriction on usage of `J1939NodeName` attributes** ⌈ A `J1939NmCluster` shall not aggregate two `J1939NmNode` s with identical `J1939NodeName` attributes.

⌋()

**[constr_3103] Range of `ecuInstance`** ⌈ The allowed values of `ecuInstance` range from 0 to 7.

⌋()

**[constr_3104] Range of `function`** ⌈ The allowed values of `function` range from 0 to 255.

⌋()

**[constr_3105] Range of `functionInstance`** ⌈ The allowed values of `functionInstance` range from 0 to 31.

⌋()

**[constr_3106] Range of `identitiyNumber`** ⌈ The allowed values of `identitiyNumber` range from 0 to 2097151.

⌋()

**[constr_3107] Range of `industryGroup`** ⌈ The allowed values of `industryGroup` range from 0 to 7.

⌋()

**[constr_3108] Range of `manufacturerCode`** ⌈ The allowed values of `manufacturerCode` range from 0 to 2047.

⌋()

**[constr_3109] Range of `vehicleSystem`** ⌈ The allowed values of `vehicleSystem` range from 0 to 127.

⌋()

**[constr_3110] Range of `vehicleSystemInstance`** ⌈ The allowed values of `vehicleSystemInstance` range from 0 to 15.

⌋()

**[constr_3111] `returnSignal` in `ClientServerToSignalMapping` is mandatory** ⌈ A `ClientServerToSignalMapping` shall always have a `returnSignal` defined.

⌋()

**[constr_3112] Invalidation support for partial mapping of a data element typed by composite data type** ⌈ If a `VariableDataPrototype` with a composite data type in a `PPortPrototype` is mapped to a `SystemSignalGroup` and only a subset of elements of the composite data type that are primitives is mapped to separate `SystemSignal` s of the `SystemSignalGroup` then at least one mapped primitive shall have an `invalidValue` defined.

⌋()

**[constr_3113] `AbstractEthernetFrame` shall not have a `PduToFrameMapping`** ⌈ It is not allowed to map `Pdu` s into `AbstractEthernetFrame` s.

⌋()

**[constr_3114] `FlatInstanceDescriptor` s pointing to the same `ParameterDataPrototype` shall have different `postBuildVariantCondition` s** ⌈ `FlatInstanceDescriptor` s that are pointing as an atpTarget to the same `ParameterDataPrototype` instance shall have different `postBuildVariantCondition` s.

⌋()

**[constr_3115] `FlatInstanceDescriptor` s pointing to the same `ParameterDataPrototype` instance** ⌈ When several `FlatInstanceDescriptor` s point to the same `ParameterDataPrototype` instance as an atpTarget in the context of a `ParameterInterface` the different `FlatInstanceDescriptor` s shall point to the `PPortPrototype` of the owning `ParameterSwComponentType` . In this case the `PPortPrototype` typed by the `ParameterInterface` is part of the context of the according `AnyInstanceRef` .

⌋()

**[constr_3116] Overlap of `ClientIdRange` s in the context of the enclosing System** ⌈ The `ClientIdRange` defined for an `EcuInstance` shall not overlap with the `ClientIdRange` of any other `EcuInstance` in the context of the enclosing System.

⌋()

**[constr_3117] Allowed value of attribute `clientId`** ⌈ Within the context of one `ClientIdDefinition` , the value of attribute `clientId` shall be in the range of `ClientIdRange` . `lowerLimit` and `ClientIdRange` . `upperLimit` for the `ClientIdRange` that is aggregated by the `EcuInstance` onto which the `SwComponentPrototype` s included in the `ClientIdDefinition` . `clientServerOperation` are mapped.

⌋()

**[constr_3118] Valid reference target for `ClientIdDefinition` . `clientServerOperation` . `contextPort`** ⌈ In the context of the definition of a `ClientIdDefinition` , the reference `clientServerOperation` . `contextPort` shall only refer to an `RPortPrototype` .

⌋()

**[constr_3121] The length of transformer chains is limited to 255 transformers** ⌈ The maximum number of `DataTransformation` . `transformerChain` references in the context of one `DataTransformation` shall be limited to 255.

⌋()

**[constr_3122] At most one transformer of each transformer class inside a transformer chain** ⌈ If the value of a `transformerClass` of a `TransformationTechnology` referenced by a `DataTransformation` does not equal `custom` , it shall be different from all other `transformerClass` values of `TransformationTechnology` s referenced by the same `DataTransformation` .

⌋()

**[constr_3123] Serializer transformer shall be the first in a chain** ⌈ A serializer transformer ( `TransformationTechnology` with attribute `transformerClass` set to `serializer` ) shall be the first transformer in a transformer chain.

⌋()

**[constr_3124] Applicability of `needsOriginalData`** ⌈ The attribute `needsOriginalData` of a `TransformationTechnology` shall only be used for the non-first transformers in the transformer chain.

⌋()

**[constr_3125] Value of attribute `inPlace` for the first transformer in a chain** ⌈ The attribute `inPlace` shall be set to |false| if the `TransformationTechnology` of the `BufferProperties` is referenced as first reference in the ordered list of references `transformerChain` from a `DataTransformation` .

⌋()

**[constr_3126] `headerLength` shall be less or equal output buffer size** ⌈ The `headerLength` shall be less or equal of the worst case output buffer size which is specified in `bufferComputation` in `BufferProperties` .

⌋()

**[constr_3127] Certain `ISignal` s always need a reference to `DataTransformation`** ⌈ An `ISignal` which references a `SystemSignal` which is referenced by a `SystemSignalGroup` in the role `transformingSystemSignal` shall always reference a `DataTransformation` .

⌋()

**[constr_3128] SOME/IP transformer configuration** ⌈ For each `TransformationDescription` variant that is a `SOMEIPTransformationDescription`

- attribute `protocol` of `TransformationTechnology` shall be set to |SOMEIP|
- attribute `version` of `TransformationTechnology` shall be set to |1.0.0|
- attribute `transformerClass` of `TransformationTechnology` shall be set to |serializer|
- attribute `headerLength` of `BufferProperties` shall be set to |64| (bits).

⌋()

**[constr_3129] Byte Order of SOME/IP transformer** ⌈ The attribute `byteOrder` of `SOMEIPTransformationDescription` shall be different from |opaque|.

⌋()

**[constr_3130] Range of Interface Version** ⌈ The value of the attribute `interfaceVersion` shall be in the range $[0; 255]$

⌋()

**[constr_3132] Required COM Based Transformation for `comBasedSignalGroupTransformation`** ⌈ If a `ISignalGroup` has a reference to the `DataTransformation` element in the role `comBasedSignalGroupTransformation` then this `DataTransformation` shall be the handled by the COM Based Transformer [13] .

⌋()

**[constr_3133] `physicalLayerType` of connected `CouplingPort` s** ⌈ The `physicalLayerType` of two `CouplingPort` s which are connected via a `CouplingPortConnection` shall be equal.

⌋()

**[constr_3134] The connection of two `CouplingPort` s with `connectionNegotiationBehavior` set to `master` is forbidden** ⌈ The `connectionNegotiation-`

Behavior of two CouplingPort s which are connected via a CouplingPortConnection shall not be both set to master .

⌋()

**[constr_3135] The connection of two CouplingPort s with connectionNegotiationBehavior set to slave is forbidden** ⌈ The connectionNegotiationBehavior of two CouplingPort s which are connected via a CouplingPortConnection shall not be both set to slave .

⌋()

**[constr_3136] Allowed payload of SecuredIPdu s** ⌈ SecuredIPdu s are allowed to reference PduTriggering s of ISignalIPdu s, ContainerIPdu s, DcmIPdu s, MultiplexedIPdu s, GeneralPurposeIPdu s with category SOMEIP_SEGMENTED_IPDU and UserDefinedIPdu s.

⌋()

**[constr_3137] IPduPort . rxSecurityVerification is configurable on the receiver side** ⌈ The IPduPort . rxSecurityVerification attribute shall only be used in IPduPort s with the communicationDirection = in.

⌋()

**[constr_3138] IPduPort . rxSecurityVerification validness** ⌈ The IPduPort . rxSecurityVerification information is only valid for SecuredIPdu s.

⌋()

**[constr_3140] No ByteOrderEnum . opaque allowed for System . containerIPduHeaderByteOrder** ⌈ The values of System . containerIPduHeaderByteOrder are restricted to ByteOrderEnum . mostSignificantByteFirst and ByteOrderEnum . mostSignificantByteLast . I.e. the value ByteOrderEnum . opaque is not allowed.

⌋()

**[constr_3141] Only IPdu s shall be part of a ContainerIPdu** ⌈ The PduTriggering which is referenced in the role ContainerIPdu . containedPduTriggering shall refer to a subclass of an IPdu in the role PduTriggering . iPdu .

⌋()

**[constr_3142] Mandatory headerIdLongHeader for longHeader** ⌈ For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu . containedPduTriggering with ContainerIPdu . headerType = longHeader the IPdu . containedIPduProps . headerIdLongHeader shall be defined.

⌋()

**[constr_3143] Mandatory headerIdShortHeader for shortHeader** ⌈ For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu . con-

tainedPduTriggering with ContainerIPdu . headerType = shortHeader the IPdu . containedIPduProps . headerIdShortHeader shall be defined.

⌋()

**[constr_3144] Mandatory IPdu . containedIPduProps for contained IPdu s** ⌈ For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu . containedPduTriggering the IPdu . containedIPduProps shall be defined.

⌋()

**[constr_3146] Partial Networking timing constraint** ⌈ For Partial Networking the following timing constraints shall be ensured:

- CAN / Ethernet: ( pnResetTime + pncPrepareSleepTimer ) < nmNetwork-Timeout
- FlexRay: ( pnResetTime + pncPrepareSleepTimer ) < nmReadySleep-Time

⌋()

**[constr_3148] executeDespiteDataUnavailability setting in case an E2E Transformer is used** ⌈ A transformer chain using E2E shall be configured with Data-Transformation . executeDespiteDataUnavailability = TRUE.

⌋()

**[constr_3149] TransformationTechnology . needsOriginalData settings for E2E Transformer** ⌈ The TransformationTechnology . needsOriginalData attribute of a TransformationTechnology element of an E2E transformer shall be set to FALSE.

⌋()

**[constr_3150] Effect of EndToEndTransformationDescription . upper-HeaderBitsToShift value in PROFILE_01 and PROFILE_11 in case it is 0** ⌈ If in PROFILE_01 or PROFILE_11 the EndToEndTransformationDescription . upperHeaderBitsToShift is equal 0 the E2E transformer used in a transformer chain with a SOME/IP transformer shall be configured with the following values:

1. EndToEndTransformationDescription . crcOffset = 0
2. EndToEndTransformationDescription . counterOffset = 8
3. For dataIdMode == lower12Bit : EndToEndTransformationDescrip-tion . dataIdNibbleOffset = 12

⌋()

**[constr_3151] BufferProperties . headerLength settings for an E2E transformer used in combination with a SOME/IP transformer** ⌈ The BufferProper-ties . headerLength for an E2E transformer located in a transformer chain with a

SOME/IP transformer shall be configured with the following values depending on the value of the `EndToEndTransformationDescription.profileName` attribute:

1. PROFILE_01: `BufferProperties.headerLength` = 16 bits
2. PROFILE_02: `BufferProperties.headerLength` = 16 bits
3. PROFILE_04: `BufferProperties.headerLength` = 96 bits
4. PROFILE_05: `BufferProperties.headerLength` = 24 bits
5. PROFILE_06: `BufferProperties.headerLength` = 40 bits
6. PROFILE_07: `BufferProperties.headerLength` = 160 bits
7. PROFILE_11: `BufferProperties.headerLength` = 16 bits
8. PROFILE_22: `BufferProperties.headerLength` = 16 bits

⌋()

**[constr_3152] `BufferProperties.headerLength` settings for an E2E transformer used in combination with a COM Based transformer** ⌈ An E2E transformer used in a transformer chain with a COM Based transformer shall be configured with the following values:

- `BufferProperties.headerLength` = 0

⌋()

**[constr_3153] E2E header field reservation required by COM Based transformer** ⌈ A COM Based transformer that is used in a transformer chain with an E2E transformer requires that the following amount of space is allocated for the E2E header fields using a proper `ISignalGroup` layout according to TPS_SYST_02068 :

**PROFILE_1:** if `dataIdMode` == `lower12Bit` : 16 bits

**PROFILE_1:** if `dataIdMode` != `lower12Bit` : 12 bits

**PROFILE_2:** 16 bits

**PROFILE_4:** 96 bits

**PROFILE_5:** 24 bits

**PROFILE_6:** 40 bits

**PROFILE_7:** 160 bits

**PROFILE_11:** if `dataIdMode` == `lower12Bit` : 16 bits

**PROFILE_11:** if `dataIdMode` == `all16Bit` : 12 bits

**PROFILE_22:** 16 bits

⌋()

**[constr_3154] `BufferProperties . bufferComputation` setting for an E2E transformer when used together with a Com-based transformer** ⌈ The `BufferProperties . bufferComputation` of an E2E transformer used in a transformer chain with a COM Based transformer shall be configured in the following way:

```
<BUFFER-COMPUTATION>
<COMPU-RATIONAL-COEFFS>
<COMPU-NUMERATOR>
 <V>0</V>
 <V>1</V>
</COMPU-NUMERATOR>
<COMPU-DENOMINATOR>
 <V>1</V>
</COMPU-DENOMINATOR>
</COMPU-RATIONAL-COEFFS>
</BUFFER-COMPUTATION>
```

⌋()

**[constr_3155] Allowed values for `EndToEndTransformationDescription . upperHeaderBitsToShift`** ⌈ The value of of the `EndToEndTransformationDescription . upperHeaderBitsToShift` attribute depends on the used serializing transformer:

**COM based transformer:** 0 (no bits are shifted)

**SOME/IP transformer:** 64 (to support the header shift of SOME/IP).

**Custom transformer:** no restriction (depends on header length and placement of custom transformer)

⌋()

**[constr_3156] Allowed values for `EndToEndTransformationISignalProps . dataId` in PROFILE_01 and PROFILE_11** ⌈ If the `EndToEndTransformationDescription . profileName` attribute has a value of PROFILE_01 or PROFILE_11 then the value of the `EndToEndTransformationISignalProps . dataId` attribute shall be in the range of 0-65535.

⌋()

**[constr_3157] Allowed values for `EndToEndTransformationISignalProps . dataId` in PROFILE_01 and PROFILE_11 in case `dataIdMode` is set to `lower12Bit`** ⌈ If the `EndToEndTransformationDescription . profileName` attribute has a value of PROFILE_01 or PROFILE_11 and the value of `EndToEndTransformationDescription . dataIdMode` attribute has a value of `lower12Bit` then the value of the `EndToEndTransformationISignalProps . dataId` attribute shall be in the range of 256-65535.

⌋()

**[constr_3158] Allowed values for `EndToEndTransformationDescription . maxDeltaCounter` in PROFILE_01 and PROFILE_11** ⌈ If the `EndToEndTrans-`

formationDescription . profileName attribute has a value of PROFILE_01 or PROFILE_11 then the attribute maxDeltaCounter shall be in the range 1-14.

⌋()

**[constr_3159] Allowed values for EndToEndTransformationDescription . maxDeltaCounter in PROFILE_04** ⌈ If the EndToEndTransformationDescription . profileName attribute has a value of PROFILE_04 the value of maxDeltaCounter attribute shall be in the range 1-65535.

⌋()

**[constr_3160] EndToEndTransformationISignalProps . dataId in PROFILE_02 and PROFILE_22** ⌈ If the EndToEndTransformationDescription . profileName attribute has a value of PROFILE_02 or PROFILE_22 then the multiplicity of the dataId attribute shall be 16 and the value of each instance shall be in the range 0..255.

⌋()

**[constr_3161] EndToEndTransformationISignalProps . dataLength in PROFILE_01, PROFILE_02, PROFILE_05, PROFILE_11, PROFILE_22** ⌈ If the EndToEndTransformationDescription . profileName attribute has a value of PROFILE_01, PROFILE_02, PROFILE_05, PROFILE_11, or PROFILE_22 then the multiplicity of the EndToEndTransformationISignalProps . dataLength attribute shall be 1.

⌋()

**[constr_3162] EndToEndTransformationISignalProps . minDataLength and EndToEndTransformationISignalProps . maxDataLength in PROFILE_01, PROFILE_02, PROFILE_05, PROFILE_11, PROFILE_22** ⌈ If the EndToEndTransformationDescription . profileName attribute has a value of PROFILE_01, PROFILE_02, PROFILE_05, PROFILE_11, or PROFILE_22 then the multiplicity of the attributes EndToEndTransformationISignalProps . minDataLength and EndToEndTransformationISignalProps . maxDataLength shall be 0.

⌋()

**[constr_3163] EndToEndTransformationISignalProps . minDataLength and EndToEndTransformationISignalProps . maxDataLength in PROFILE_04, PROFILE_06, PROFILE_07** ⌈ If the EndToEndTransformationDescription . profileName attribute has a value of PROFILE_04, PROFILE_06, or PROFILE_07 then the multiplicity of the attributes EndToEndTransformationISignalProps . minDataLength and EndToEndTransformationISignalProps . maxDataLength shall be 1.

⌋()

**[constr_3164] EndToEndTransformationISignalProps . dataLength in PROFILE_04, PROFILE_06, PROFILE_07** ⌈ If the EndToEndTransformationDescription . profileName attribute has a value of PROFILE_04, PROFILE_06, or

PROFILE_07 then the multiplicity of the attribute `EndToEndTransformationISignalProps.dataLength` shall be 0.

⌋()

**[constr_3165] Effect of `EndToEndTransformationDescription.upperHeaderBitsToShift` value in PROFILE_01, PROFILE_11** ⌈ If the `EndToEndTransformationDescription.profileName` attribute has a value of PROFILE_01 or PROFILE_11 then:

1. `EndToEndTransformationDescription.crcOffset` shall be set to the same value of `upperHeaderBitsToShift`.
2. `EndToEndTransformationDescription.counterOffset` shall be set to the value of `upperHeaderBitsToShift` + 8.
3. (if used) `EndToEndTransformationDescription.dataIdNibbleOffset` shall be set to the value of `upperHeaderBitsToShift` + 12.

⌋()

**[constr_3166] `EndToEndTransformationDescription.upperHeaderBitsToShift` in PROFILE_02** ⌈ If the `EndToEndTransformationDescription.profileName` attribute has a value of PROFILE_02 then the value of the `upperHeaderBitsToShift` attribute shall be 0.

⌋()

**[constr_3167] Effect of `EndToEndTransformationDescription.upperHeaderBitsToShift` value in PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07** ⌈ If the `EndToEndTransformationDescription.profileName` attribute has a value of PROFILE_04, PROFILE_05, PROFILE_06, or PROFILE_07 the value of the `EndToEndTransformationDescription.offset` attribute shall be equal to the value of the `EndToEndTransformationDescription.upperHeaderBitsToShift` attribute.

⌋()

**[constr_3169] Attribute multiplicities and values in PROFILE_02 and PROFILE_22** ⌈ If the `EndToEndTransformationDescription.profileName` attribute has a value of PROFILE_02 or PROFILE_22 then:

1. the multiplicity of the `EndToEndTransformationDescription.crcOffset` attribute shall be 0.
2. the multiplicity of the `EndToEndTransformationDescription.counterOffset` attribute shall be 0.
3. the multiplicity of the `EndToEndTransformationDescription.dataIdNibbleOffset` attribute shall be 0.
4. the value of the `EndToEndTransformationDescription.offset` attribute shall be 0.

⌋()

**[constr_3171] Value of `EndToEndTransformationISignalProps . dataId` shall be unique in PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07** ⌈ If the `EndToEndTransformationDescription . profileName` attribute has a value of PROFILE_04, PROFILE_05, PROFILE_06, or PROFILE_07 then the value of the `EndToEndTransformationISignalProps . dataId` attribute shall be unique within the scope of the `System` .

⌋()

**[constr_3172] Effect of `EndToEndTransformationDescription . profileBehavior` value in PROFILE_01** ⌈ If the `EndToEndTransformationDescription . profileName` attribute has a value of PROFILE_01 and the value of the `profileBehavior` attribute is `R4_2` then:

- the value of the `EndToEndTransformationDescription . maxNoNewOrRepeatedData` attribute shall be 14.
- the value of the `EndToEndTransformationDescription . syncCounterInit` attribute shall be 1.

⌋()

**[constr_3173] Effect of `EndToEndTransformationDescription . profileBehavior` value in PROFILE_02** ⌈ If the `EndToEndTransformationDescription . profileName` attribute has a value of PROFILE_02 and the value of the `profileBehavior` attribute is `R4_2` then:

- the value of the `EndToEndTransformationDescription . maxNoNewOrRepeatedData` attribute shall be 15.
- the value of the `EndToEndTransformationDescription . syncCounterInit` attribute shall be 1.

⌋()

**[constr_3174] `EndToEndTransformationDescription` settings not allowed in PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, PROFILE_11, PROFILE_22** ⌈ If the `EndToEndTransformationDescription . profileName` attribute has a value of PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, PROFILE_11, or PROFILE_22 then:

1. the multiplicity of the `EndToEndTransformationDescription . maxNoNewOrRepeatedData` attribute shall be 0.
2. the multiplicity of the `EndToEndTransformationDescription . syncCounterInit` attribute shall be 0.
3. the multiplicity of the `EndToEndTransformationDescription . profileBehavior` attribute shall be 0.

⌋()

**[constr_3176] Value range of `windowSize`** ⌈ The value of the `windowSize` attribute shall be greater or equal to 1.

⌋()

**[constr_3177] Dependency between `maxErrorStateValid` , `maxErrorStateInit` and `maxErrorStateInvalid`** ⌈ The following restriction shall be respected:
`maxErrorStateValid` >= `maxErrorStateInit` >= `maxErrorStateInvalid` >= 0

⌋()

**[constr_3178] Dependency between `minOkStateValid` , `minOkStateInit` and `minOkStateInvalid`** ⌈ The following restriction shall be respected:
1 <= `minOkStateValid` <= `minOkStateInit` <= `minOkStateInvalid`

⌋()

**[constr_3179] Dependency between `minOkStateInit` , `maxErrorStateInit` and `windowSize`** ⌈ The following restriction shall be respected:
`minOkStateInit` + `maxErrorStateInit` <= `windowSize`

⌋()

**[constr_3180] Dependency between `minOkStateValid` , `maxErrorStateValid` and `windowSize`** ⌈ The following restriction shall be respected:
`minOkStateValid` + `maxErrorStateValid` <= `windowSize`

⌋()

**[constr_3181] Dependency between `minOkStateInvalid` , `maxErrorStateInvalid` and `windowSize`** ⌈ The following restriction shall be respected: `minOkStateInvalid` + `maxErrorStateInvalid` <= `windowSize`

⌋()

**[constr_3182] Restriction on `TransformationTechnology` . `transformationDescription` `VariationPoint`** ⌈ The `EndToEndTransformationDescription` . `profileName` attribute shall not be subject to variability for a given `ISignal` / `ISignalGroup` , i.e., the value of the `EndToEndTransformationDescription` . `profileName` attribute shall be the same in all different variants.

⌋()

**[constr_3183] `ISignalGroup` with `transformationISignalProps`** ⌈ An `ISignalGroup` that aggregates `transformationISignalProps` shall reference the `DataTransformation` in the role `comBasedSignalGroupTransformation` .

⌋()

**[constr_3184] Only one `EndToEndTransformationISignalProps` . `dataId` element in PROFILE_01 and PROFILE_11** ⌈ If the `EndToEndTransformationDe-`

scription . profileName attribute has a value of PROFILE_01 or PROFILE_11 then the multiplicity of the EndToEndTransformationISignalProps . dataId attribute shall be 1.

⌋()

**[constr_3185] Multiplicity of EndToEndTransformationDescription . dataIdMode in PROFILE_01 and PROFILE_11** ⌈ If the EndToEndTransformationDescription . profileName attribute is set to PROFILE_01 or PROFILE_11 then the multiplicity of the EndToEndTransformationDescription . dataIdMode attribute shall be 1.

⌋()

**[constr_3186] Multiplicity of EndToEndTransformationDescription . dataIdMode in PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, PROFILE_22** ⌈ If the EndToEndTransformationDescription . profileName attribute is set to a value of PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, or PROFILE_22 then the multiplicity of the EndToEndTransformationDescription . dataIdMode attribute shall be 0.

⌋()

**[constr_3187] Multiplicity of EndToEndTransformationDescription . counterOffset in PROFILE_01 and PROFILE_11** ⌈ If the EndToEndTransformationDescription . profileName attribute is set to PROFILE_01 or PROFILE_11 then the multiplicity of the EndToEndTransformationDescription . counterOffset attribute shall be 1.

⌋()

**[constr_3188] Multiplicity of EndToEndTransformationDescription . counterOffset in PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, PROFILE_22** ⌈ If the EndToEndTransformationDescription . profileName attribute is set to a value of PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, or PROFILE_22 then the multiplicity of the EndToEndTransformationDescription . counterOffset attribute shall be 0.

⌋()

**[constr_3189] Multiplicity of EndToEndTransformationDescription . crcOffset in PROFILE_01 and PROFILE_11** ⌈ If the EndToEndTransformationDescription . profileName attribute is set to PROFILE_01 or PROFILE_11 then the multiplicity of the EndToEndTransformationDescription . crcOffset attribute shall be 1.

⌋()

**[constr_3190] Multiplicity of EndToEndTransformationDescription . crcOffset in PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, PROFILE_22** ⌈ If the EndToEndTransformationDescription . profileName attribute is set to a value of PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06,

PROFILE_07, or PROFILE_22 then the multiplicity of the EndToEndTransformationDescription.crcOffset attribute shall be 0.

⌋()

**[constr_3191] Multiplicity of EndToEndTransformationDescription.dataIdNibbleOffset in PROFILE_01, PROFILE_11 and dataIdMode equal to lower12Bit** ⌈ If the EndToEndTransformationDescription.profileName attribute is set to PROFILE_01 or PROFILE_11 and the value of the EndToEndTransformationDescription.dataIdMode attribute is set to lower12Bit then the multiplicity of the EndToEndTransformationDescription.dataIdNibbleOffset attribute shall be 1.

⌋()

**[constr_3192] Multiplicity of EndToEndTransformationDescription.dataIdNibbleOffset in PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, PROFILE_22 or dataIdMode different from lower12Bit** ⌈ If the EndToEndTransformationDescription.profileName attribute is set to a value of PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, or PROFILE_22 or the EndToEndTransformationDescription.dataIdMode attribute is set to value different from lower12Bit then the multiplicity of the EndToEndTransformationDescription.dataIdNibbleOffset attribute shall be 0.

⌋()

**[constr_3193] Multiplicity of EndToEndTransformationDescription.offset in PROFILE_01 and PROFILE_11** ⌈ If the EndToEndTransformationDescription.profileName attribute is set to PROFILE_01 or PROFILE_11 then the multiplicity of the EndToEndTransformationDescription.offset attribute shall be 0.

⌋()

**[constr_3194] Multiplicity of EndToEndTransformationDescription.offset in PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, PROFILE_22** ⌈ If the EndToEndTransformationDescription.profileName attribute is set to a value PROFILE_02, PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, or PROFILE_22 then the multiplicity of the EndToEndTransformationDescription.offset attribute shall be 1.

⌋()

**[constr_3195] Allowed values for EndToEndTransformationDescription.maxDeltaCounter in PROFILE_02 and PROFILE_22** ⌈ If the EndToEndTransformationDescription.profileName attribute has a value of PROFILE_02 or PROFILE_22 then the attribute maxDeltaCounter shall be in the range 1-15.

⌋()

**[constr_3196] Allowed values for `EndToEndTransformationDescription.maxDeltaCounter` in PROFILE_05** ⌈ If the `EndToEndTransformationDescription.profileName` attribute has a value of PROFILE_05 then the attribute `maxDeltaCounter` shall be in the range 1-255.

⌋()

**[constr_3197] Allowed values for `EndToEndTransformationDescription.maxDeltaCounter` in PROFILE_06** ⌈ If the `EndToEndTransformationDescription.profileName` attribute has a value of PROFILE_06 then the attribute `maxDeltaCounter` shall be in the range 1-255.

⌋()

**[constr_3198] Uniqueness of `PncMapping.shortLabel`** ⌈ If the optional `shortLabel` attribute is used it shall be unique in the `System` scope.

⌋()

**[constr_3199] `ISignal` that has `dataTypePolicy` set to `transformingISignal` shall reference a `DataTransformation`** ⌈ In a complete model every `ISignal` that has `dataTypePolicy` set to `transformingISignal` shall reference a `DataTransformation`.

⌋()

**[constr_3201] `eventGroupIdentifier` in `ConsumedEventGroup`s that are referenced by the same `EventHandler`** ⌈ In case that an `EventHandler` refers to several `ConsumedEventGroup`s all these `ConsumedEventGroup`s shall have the same `eventGroupIdentifier`.

⌋()

**[constr_3202] `LinFrameTriggering` to `LinUnconditionalFrame` reference restriction in `LinEventTriggeredFrame` context** ⌈ Within a `PhysicalChannel` a `LinUnconditionalFrame` shall be referenced by only one `LinFrameTriggering` to allow a derivation of the identifier of a substituted Frame if the `LinUnconditionalFrame` is referenced by a `LinEventTriggeredFrame` in the role `linUnconditionalFrame`.

⌋()

**[constr_3203] `LinFrameTriggering` to `LinSporadicFrame` reference restriction in `LinSporadicFrame` context** ⌈ Within a `PhysicalChannel` a `LinUnconditionalFrame` shall be referenced by only one `LinFrameTriggering` to allow a derivation of the identifier of a substituted Frame if the `LinUnconditionalFrame` is referenced by a `LinSporadicFrame` in the role `substitutedFrame`.

⌋()

**[constr_3204] `LinUnconditionalFrame`s associated with a `LinSporadicFrame`** ⌈ A `LinUnconditionalFrame` associated with a `LinSpo-`

radicFrame shall not be allocated in the same LinScheduleTable as the LinSporadicFrame .

⌋()

**[constr_3205] Existence of FramePort for a FrameTriggering that references a LinSporadicFrame** ⌈ A FrameTriggering that references a LinSporadicFrame shall not have a reference to a FramePort .

⌋()

**[constr_3206] Existence of FramePort for a FrameTriggering that references a LinEventTriggeredFrame** ⌈ A FrameTriggering that references a LinEventTriggeredFrame shall not have a reference to a FramePort .

⌋()

**[constr_3208] executeDespiteDataUnavailability usage restriction** ⌈ In the set of more than one ISignal which reference the same SystemSignal in the role systemSignal , there shall be no ISignal which references a DataTransformation where executeDespiteDataUnavailability is set to true.

⌋()

**[constr_3209] CanFrameTriggering s with identical PGN** ⌈ For all CanFrameTriggering s where the attribute identifier contains the identical PGN (as defined in section 5.2 Protocol Data Unit in [14] ) the attribute j1939requestable shall also have an identical value.

⌋()

**[constr_3210] J1939TpPg s with identical pgn value** ⌈ For all J1939TpPg s where the attribute pgn has an identical value the attribute requestable shall also have an identical value.

⌋()

**[constr_3211] PduTriggering s with triggerIPduSendCondition** ⌈ Only PduTriggering s with references to ISignalIPdu s are allowed to contain a triggerIPduSendCondition .

⌋()

**[constr_3212] Limitation of DoIpTpConnection.tpSdu** ⌈ DoIpTpConnection shall only reference PduTriggering s of DcmIPdu s in the role tpSdu .

⌋()

**[constr_3213] TransformationISignalProps . csErrorReaction setting in case that the serializer transformerClass and Client/Server communication is used** ⌈ In TransformationISignalProps the attribute csErrorReaction shall be set if the TransformationISignalProps specifies the details for a TransformationTechnology with transformerClass equal to serializer

and the ISignal that aggregates the TransformationISignalProps transports a client/server communication.

⌋()

**[constr_3214] TransformationISignalProps . csErrorReaction setting in case that a transformerClass different from serializer is used or the Client/Server communication is not used** ⌈ In TransformationISignalProps the attribute csErrorReaction shall not be used if the TransformationISignalProps specifies the details for a TransformationTechnology with transformerClass not equal to serializer or the ISignal that aggregates the TransformationISignalProps does not transport a client/server communication.

⌋()

**[constr_3215] TransformationTechnology . version and TransformationTechnology . protocol settings for request and response of a client/server communication** ⌈ TransformationTechnology . version and TransformationTechnology . protocol shall be identical for ISignal s that are derived from the same ClientServerOperation . This means that all ISignal s that refer to ClientServerToSignalMapping . callSignal or to ClientServerToSignalMapping . returnSignal of the same ClientServerToSignalMapping shall have the same TransformationTechnology . protocol and TransformationTechnology . version defined.

⌋()

**[constr_3216] Usage of SOMEIPTransformationISignalProps . sessionHandlingSR** ⌈ The attribute sessionHandlingSR of SOMEIPTransformationISignalProps shall only be used for ISignal s which reference SystemSignal s which are mapped via a SenderReceiverToSignalMapping .

⌋()

**[constr_3218] Range of Size of Fixed-size Array Length Fields** ⌈ The value of attribute sizeOfArrayLengthFields of SOMEIPTransformationISignalProps shall be either 0, 1, 2 or 4.

⌋()

**[constr_3219] The existence of LinSlave s in the LinMaster EcuExtract** ⌈ LinSlave s shall not be part of the EcuExtract of the corresponding LinMaster .

⌋()

**[constr_3220] Range of Size of Structure Length Fields** ⌈ The value of attribute sizeOfStructLengthFields of SOMEIPTransformationISignalProps shall be either 0, 1, 2 or 4.

⌋()

**[constr_3221] Range of Size of Union Length Fields** ⌈ The value of attribute `sizeOfUnionLengthFields` of `SOMEIPTransformationISignalProps` shall be either 0, 1, 2 or 4.

⌋()

**[constr_3222] No `ByteOrderEnum.opaque` allowed for `PduToFrameMapping.packingByteOrder`** ⌈ The values of `PduToFrameMapping.packingByteOrder` are restricted to `ByteOrderEnum.mostSignificantByteFirst` and `ByteOrderEnum.mostSignificantByteLast`. I.e. the value `ByteOrderEnum.opaque` is not allowed.

⌋()

**[constr_3223] No `ByteOrderEnum.opaque` allowed for `MultiplexedIPdu.selectorFieldByteOrder`** ⌈ The values of `MultiplexedIPdu.selectorFieldByteOrder` are restricted to `ByteOrderEnum.mostSignificantByteFirst` and `ByteOrderEnum.mostSignificantByteLast`. I.e. the value `ByteOrderEnum.opaque` is not allowed.

⌋()

**[constr_3224] No `ByteOrderEnum.opaque` allowed for `SegmentPosition.segmentByteOrder`.** ⌈ The values of `SegmentPosition.segmentByteOrder` are restricted to `ByteOrderEnum.mostSignificantByteFirst` and `ByteOrderEnum.mostSignificantByteLast`. I.e. the value `ByteOrderEnum.opaque` is not allowed.

⌋()

**[constr_3225] `LinFrameTriggering.linChecksum` not allowed for `LinSporadicFrame`s** ⌈ The `linChecksum` attribute of a `LinFrameTriggering` that references a `LinSporadicFrame` shall not be set.

⌋()

**[constr_3226] `LinFrameTriggering.linChecksum` for `LinEventTriggeredFrame`s** ⌈ Within a `PhysicalChannel` the `linChecksum` attribute of a `LinFrameTriggering` that references a `LinEventTriggeredFrame` shall have the same value as the `linChecksum` attribute of each `LinFrameTriggering` that references a `LinUnconditionalFrame` that in turn is referenced by that `LinEventTriggeredFrame`.

⌋()

**[constr_3227] `NmNode.nmPassiveModeEnabled` setting** ⌈ `NmNode.nmPassiveModeEnabled` shall be set to the same value in all `NmCluster`s with the same bus protocol in the scope of one `NmEcu`.

⌋()

**[constr_3229] `SwComponentPrototype` mapped to an `ApplicationPartition` and `EcuInstance`** ⌈ If the `SwcToEcuMapping.ecuInstance` exists then a `SwComponentPrototype` that is mapped to an `ApplicationPartition` via the `SwcToApplicationPartitionMapping` shall only be mapped by an `ApplicationPartitionToEcuPartitionMapping` to an `EcuPartition` that is aggregated by the `EcuInstance` referenced by means of `SwcToEcuMapping.ecuInstance`.

⌋()

**[constr_3230] Usage of `SenderRecRecordElementMapping.applicationRecordElement`** ⌈ `SenderRecRecordElementMapping.applicationRecordElement` shall only be used if the referenced context element ( `VariableDataPrototype` that is referenced by the `SenderReceiverToSignalGroupMapping.dataElement` ) is typed by an `ApplicationDataType`.

⌋()

**[constr_3231] Usage of `IndexedArrayElement.applicationArrayElement`** ⌈ `IndexedArrayElement.applicationArrayElement` shall only be used if the referenced context element ( `VariableDataPrototype` that is referenced by the `SenderReceiverToSignalGroupMapping.dataElement` ) is typed by an `ApplicationDataType`.

⌋()

**[constr_3232] `ApplicationPartition` is allowed to be mapped to only one `EcuPartition`** ⌈ Each `ApplicationPartition` shall be mapped at most once to an `EcuPartition` via the `ApplicationPartitionToEcuPartitionMapping`.

⌋()

**[constr_3239] Consistent mapping of software-component to `J1939NmNode`** ⌈ The value of attribute `J1939NmNode.nodeName.function` of a `J1939NmNode` referenced by `J1939ControllerApplicationToJ1939NmNodeMapping` in the role `j1939NmNode` shall be identical to the value of `J1939ControllerApplication.functionId`.

⌋()

**[constr_3240] Consistent mapping of `J1939ControllerApplication` to `EcuInstance`** ⌈ A `SwComponentPrototype` that is referenced by a `J1939ControllerApplication` mapped to a specific `J1939NmNode` shall only be mapped to an `EcuInstance` that in turn owns the same `J1939NmNode`.

⌋()

**[constr_3241] Usage of `AssignFrameId.messageId`** ⌈ The value of `AssignFrameId.messageId` for the `AssignFrameId` that refers to a `LinSlave` in the role `assignedController` shall be equal to the `messageId` of the `LinConfigurableFrame` aggregated by `LinCommunicationConnector` in role `linConfigurableFrame` that points to this `LinSlave` in the role `commController`.

⌋()

**[constr_3242] Usage of `UnassignFrameId . messageId`** ⌈ The value of `UnassignFrameId . messageId` for the `UnassignFrameId` that refers to a `LinSlave` in the role `assignedController` shall be equal to the `messageId` of the `LinConfigurableFrame` aggregated by `LinCommunicationConnector` in role `linConfigurableFrame` that points to this `LinSlave` in the role `commController` .

⌋()

**[constr_3243] `FrameTriggering . pduTriggering` condition** ⌈ A `FrameTriggering` shall reference a `PduTriggering` if the `PduTriggering` references a `Pdu` that is referenced by a `PduToFrameMapping` which in turn is aggregated by the `Frame` that is referenced by that `FrameTriggering` .

⌋()

**[constr_3244] Usage of `SenderRecRecordElementMapping . implementationRecordElement`** ⌈ `SenderRecRecordElementMapping . implementationRecordElement` shall only be used if the referenced context element ( `VariableDataPrototype` that is referenced by the `SenderReceiverToSignalGroupMapping . dataElement` ) is typed by an `ImplementationDataType` .

⌋()

**[constr_3245] Usage of `IndexedArrayElement . implementationArrayElement`** ⌈ `IndexedArrayElement . implementationArrayElement` shall only be used if the referenced context element ( `VariableDataPrototype` that is referenced by the `SenderReceiverToSignalGroupMapping . dataElement` ) is typed by an `ImplementationDataType` .

⌋()

**[constr_3246] `Frame . packingByteOrder` mix within a `Frame` is not allowed** ⌈ All `PduToFrameMapping` s within a `Frame` shall have the same `packingByteOrder` value.

⌋()

**[constr_3247] Byte order mix within a `MultiplexedIPdu` is not allowed** ⌈ The `segmentByteOrder` of all `SegmentPosition` s and the `selectorFieldByteOrder` shall have the same value in the `MultiplexedIPdu` .

⌋()

**[constr_3248] Category of `HwElement` for `ECUMapping`** ⌈ The `HwElement` which is referenced from `ECUMapping` in the role `ecu` shall be of category `MicroController`

⌋()

**[constr_3249] Category of `HwElement` for `SwcToEcuMapping`** ⌈ The `HwElement` which is referenced from `SwcToEcuMapping` in the role `processingUnit` shall be of category "ProcessingUnit".

⌋()

**[constr_3250] `PduTriggering.iSignalTriggering` condition** ⌈ A `PduTriggering` shall reference an `ISignalTriggering` if the `ISignalTriggering` references an `ISignal` or an `ISignalGroup` that is referenced by an `ISignalToIPduMapping` which in turn is aggregated by the `Pdu` that is referenced by that `PduTriggering`.

⌋()

**[constr_3251] Value of `GlobalTimeDomain.domainId` in `subDomain` chains** ⌈ In a chain of `GlobalTimeDomain.subDomain` the value of the attribute `GlobalTimeDomain.domainId` shall be identical.

⌋()

**[constr_3252] `ISignalTriggering.iSignalPort` reference condition** ⌈ An `ISignalTriggering` shall only reference an `ISignalPort` if the `CommunicationConnector` aggregating that `ISignalPort` is referenced by the `PhysicalChannel` which in turn aggregates that `ISignalTriggering`.

⌋()

**[constr_3253] `PduTriggering.iPduPort` reference condition** ⌈ A `PduTriggering` shall only reference an `IPduPort` if the `CommunicationConnector` aggregating that `IPduPort` is referenced by the `PhysicalChannel` which in turn aggregates that `PduTriggering`.

⌋()

**[constr_3254] `FrameTriggering.framePort` reference condition** ⌈ A `FrameTriggering` shall only reference a `FramePort` if the `CommunicationConnector` aggregating that `FramePort` is referenced by the `PhysicalChannel` which in turn aggregates that `FrameTriggering`.

⌋()

**[constr_3255] `FrameTriggering.pduTriggering` reference condition with regard to the `PhysicalChannel`** ⌈ A `FrameTriggering` shall only reference a `PduTriggering` in the role `pduTriggering` if both the `FrameTriggering` and `PduTriggering` are aggregated by the same `PhysicalChannel`.

⌋()

**[constr_3256] `PduTriggering.iSignalTriggering` reference condition with regard to the `PhysicalChannel`** ⌈ A `PduTriggering` shall only reference an `ISignalTriggering` in the role `iSignalTriggering` if both the `PduTriggering` and `ISignalTriggering` are aggregated by the same `PhysicalChannel`.

⌋()

**[constr_3257] TimeSyncTechnology of servers and clients in a time synchronized network.** ⌈ `TimeSyncClientConfiguration.timeSyncTechnology` shall

have the same value as the TimeSyncServerConfiguration . timeSyncTechnology that is referenced in the TimeSyncClientConfiguration . orderedMaster list.

⌋()

**[constr_3258] Restriction on ISignal . length in case iSignalType is set to array** ⌈ If ISignal . iSignalType is set to array then ISignal . length shall be a multiple of 8.

⌋()

**[constr_3259] Allowed use of SdServerConfig . capabilityRecord** ⌈ A TagWithOptionalValue element may only be composed (in role capabilityRecord ) by a SdServerConfig element if the respective SdServerConfig element is directly composed by a ProvidedServiceInstance element in role sdServerConfig . A TagWithOptionalValue element must not be composed (in role capabilityRecord ) by an SdServerConfig element if the respective SdServerConfig element is composed by an EventHandler element in role sdServerConfig .

⌋()

**[constr_3260] Allowed use of SdClientConfig . capabilityRecord** ⌈ A TagWithOptionalValue element may only be composed (in role capabilityRecord ) by a SdClientConfig element if the respective SdClientConfig element is directly composed by a ConsumedServiceInstance element in role sdClientConfig . A TagWithOptionalValue element must not be composed (in role capabilityRecord ) by an SdClientConfig element if the respective SdClientConfig element is composed by a ConsumedEventGroup element in role sdClientConfig .

⌋()

**[constr_3261] GlobalTimeDomain . globalTimePduTriggering category** ⌈ The Pdu that is referenced by the PduTriggering that in turn is referenced by GlobalTimeDomain in the role globalTimePduTriggering shall be a GeneralPurposePdu of category GLOBAL_TIME.

⌋()

**[constr_3262] ConsumedEventGroup . eventGroupIdentifier is mandatory** ⌈ The ConsumedEventGroup . eventGroupIdentifier is mandatory.

⌋()

**[constr_3263] Restriction of usage of SwcToEcuMapping in a System** ⌈ For all SwcToEcuMapping s in a System the following restriction applies: No two SwcToEcuMapping s shall have the exact same reference to

- SwComponentPrototype
- EcuInstance

- processingUnit
- controlledHwElement

⌋()

**[constr_3264] Server side ClientServerToSignalMapping s in case of a n:1 inter-ECU client-server communication** ⌈ If within the System with category SYSTEM_DESCRIPTION or SYSTEM_EXTRACT the ClientServerToSignalMapping s for inter-ECU n:1 client-server communication are placed on the provider (server) side, then each of these ClientServerToSignalMapping s shall (in the hierarchy of SwComponentPrototype s) refer to a "unique communication path" w.r.t. the EcuInstance s the client SwComponentPrototype s are mapped to.

⌋()

**[constr_3265] TransformationTechnology . hasInternalState setting for an E2E transformer** ⌈ The value of hasInternalState shall be set to true for a TransformationTechnology with transformerClass set to safety .

⌋()

**[constr_3266] TransformationTechnology . hasInternalState setting for a SOME/IP Transformer** ⌈ The value of hasInternalState shall be set to true for a SOME/IP Transformer if SOMEIPTransformationISignalProps . sessionHandlingSR for the ISignal is set to active.

⌋()

**[constr_3267] PduTriggering s in Service Discovery SocketConnectionBundle s** ⌈ SD SocketConnectionBundle s defined in TPS_SYST_02119 shall only refer to PduTriggering s which point to GeneralPurposePdu s of category SD.

⌋()

**[constr_3268] Service Discovery SocketConnectionBundle serverPort reference to a TpPort** ⌈ Each SD SocketConnectionBundle defined in TPS_SYST_02119 shall refer with the serverPort reference to an ApplicationEndpoint (via SocketAddress ) with a Udp Port.

⌋()

**[constr_3269] Service Discovery SocketConnection clientPort reference to a TpPort** ⌈ Each SD SocketConnection defined in TPS_SYST_02119 shall refer with the clientPort reference to an ApplicationEndpoint (via SocketAddress ) with Udp Port dynamicallyAssigned set to true.

⌋()

**[constr_3270] Service Discovery SocketConnection clientPort reference to an IP Address** ⌈ Each SD SocketConnection defined in TPS_SYST_02119 shall refer with the clientPort reference to a NetworkEndpoint (via SocketAddress . applicationEndpoint ) with IP Address ANY (IPv4 or IPv6).

⌋()

**[constr_3271] `clientIpAddrFromConnectionRequest` and `clientPortFromConnectionRequest` settings for SD `SocketConnection` s** ⌈ SD `SocketConnection` s defined in TPS_SYST_02119 shall define `clientIpAddrFromConnectionRequest` set to true and `clientPortFromConnectionRequest` set to true.

⌋()

**[constr_3272] `SocketConnectionIpduIdentifier . headerId` setting for SD `SocketConnectionBundle` s** ⌈ The `SocketConnectionIpduIdentifier . headerId` of SD `SocketConnectionBundle` s defined in TPS_SYST_02119 shall always be set to 0xFFFF8100 for SD messages.

⌋()

**[constr_3273] Service Discovery multicast `SocketConnectionBundle` 's `serverPort` reference to an IP Address** ⌈ The SD `SocketConnectionBundle` for multicast defined in TPS_SYST_02119 ( `SocketConnectionBundle` B) shall refer via the `serverPort` to a `SocketAddress` representing a Multicast Address.

⌋()

**[constr_3274] Service Discovery unicast `SocketConnectionBundle` 's `serverPort` reference to an IP Address** ⌈ The SD `SocketConnectionBundle` for unicast defined in TPS_SYST_02119 ( `SocketConnectionBundle` A) shall refer via the `serverPort` to a `SocketAddress` representing a Unicast Address.

⌋()

**[constr_3275] `PduTriggering` containment in different `PdurIPduGroup` s of the same `EcuInstance` is not allowed** ⌈ A `PduTriggering` shall not be referenced by more than one `PdurIPduGroup` in the role `iPdu` where each of these `PdurIPduGroup` s are referenced by the same `EcuInstance` .

⌋()

**[constr_3276] Prohibition of usage of `allowedIPv6ExtHeaders` in IPv4 `SocketConnection` s** ⌈ IPv4 `SocketConnection` s shall not define `allowedIPv6ExtHeaders` . An IPv4 `SocketConnection` points to a `SocketAddress` in the role `clientPort` and relates to an `ApplicationEndpoint` that refers to a `NetworkEndpoint` that has an `Ipv4Configuration` as `networkEndpointAddress` .

⌋()

**[constr_3277] Restriction of usage of `IPv6ExtHeaderFilterList` s in IPv6 `SocketConnection` s** ⌈ All `SocketConnection` s related to the same IPv6 `NetworkEndpoint` shall all reference either no or exactly the same `IPv6ExtHeaderFilterList` with the `allowedIPv6ExtHeaders` attribute.

⌋()

**[constr_3278] Usage of `SOMEIPTransformationProps.sizeOfArrayLengthField`** ⌈ The attribute `sizeOfArrayLengthField` of `SOMEIPTransformationProps` shall only be defined if the `DataPrototypeTransformationProps` is defined for a static size array according to TPS_SYST_02121 .

⌋()

**[constr_3279] Usage of `SOMEIPTransformationProps.sizeOfStructLengthField`** ⌈ The attribute `sizeOfStructLengthField` of `SOMEIPTransformationProps` shall only be defined if the `DataPrototypeTransformationProps` is defined for a structure according to TPS_SYST_02121 .

⌋()

**[constr_3280] Usage of `SOMEIPTransformationProps.sizeOfUnionLengthField`** ⌈ The attribute `sizeOfUnionLengthField` of `SOMEIPTransformationProps` shall only be defined if the `DataPrototypeTransformationProps` is defined for a union according to TPS_SYST_02121 .

⌋()

**[constr_3281] Usage of `SOMEIPTransformationProps.alignment`** ⌈ The attribute `alignment` of `SOMEIPTransformationProps` shall only be defined if the `DataPrototypeTransformationProps` is defined for a variable data length data element according to TPS_SYST_02121 .

⌋()

**[constr_3282] SOME/IP Transformation settings for static size arrays in the context of an `ISignal`** ⌈ In the context of an `ISignal` the usage of `DataPrototypeTransformationProps.transformationProps.sizeOfArrayLengthField` is only allowed if the `SOMEIPTransformationISignalProps.sizeOfArrayLengthFields` is not defined.

⌋()

**[constr_3283] SOME/IP Transformation settings for structures in the context of an `ISignal`** ⌈ In the context of an `ISignal` the usage of `DataPrototypeTransformationProps.transformationProps.sizeOfStructLengthField` is only allowed if the `SOMEIPTransformationISignalProps.sizeOfStructLengthFields` is not defined.

⌋()

**[constr_3284] SOME/IP Transformation settings for unions in the context of an `ISignal`** ⌈ In the context of an `ISignal` the usage of `DataPrototypeTransformationProps.transformationProps.sizeOfUnionLengthField` is only allowed if the `SOMEIPTransformationISignalProps.sizeOfUnionLengthFields` is not defined.

⌋()

**[constr_3285] Alignment of variable data length data elements in the context of an `ISignal`** ⌈ The definition of `DataPrototypeTransformationProps.transformationProps.alignment` is only allowed if the `SOMEIPTransformationDescription.alignment` is not defined.

⌋()

**[constr_3286] `ISignal.length` shall be consistent to transformer configuration** ⌈ For `ISignal` s that are used for transformed data, the value `ISignal.length` shall be greater or equal to the maximum possible size of the transformed data (including alignment). This size can be calculated by using the formulas specified in the `TransformationTechnology.bufferProperties.bufferComputation` of all TransformationTechnologies in the ordered list `DataTransformation.transformerChain` for the length that is determined from the mapped `VariableDataPrototype`.

⌋()

**[constr_3297] Prohibition of usage of `allowedTcpOptions` in Udp `SocketConnection` s** ⌈ Udp `SocketConnection` s shall not define `allowedTcpOptions`. A Udp `SocketConnection` points to a `SocketAddress` in the role `clientPort` and relates to an `ApplicationEndpoint` that has a `UdpTp` defined as `tpConfiguration`.

⌋()

**[constr_3298] `Ipv6Configuration.ipv6Address` range in case of `enableAnycast`** ⌈ If `Ipv6Configuration.enableAnycast` is set to true then the `Ipv6Configuration.ipv6Address` needs to be in the unicast addressing range.

⌋()

**[constr_3299] `SocketConnectionBundle.pathMtuDiscoveryEnabled` setting dependency** ⌈ `SocketConnectionBundle.pathMtuDiscoveryEnabled` shall only be set to TRUE if `EthernetCommunicationConnector.pathMtuEnabled` == TRUE.

⌋()

**[constr_3311] Usage of `SocketConnectionBundle.flowLabel`** ⌈ `SocketConnectionBundle.flowLabel` shall only be used if the `SocketConnectionBundle` points to a `SocketAddress` in the role `serverPort` with an `ApplicationEndpoint` that refers to a `NetworkEndpoint` with an `Ipv6Configuration`.

⌋()

**[constr_3312] Consistency of `vlanPriority` and `EthernetCommunicationConnector`** ⌈ A `GlobalTimeEthMaster` refers to an `EthernetCommunicationConnector` in the role `communicationConnector`. If that `EthernetCommunicationConnector` is referenced by an `EthernetPhysicalChannel` in the role `commConnector` and the `EthernetPhysicalChannel` has a vLan tag defined via

the VlanConfig then the GlobalTimeEthMaster shall have a vlanPriority defined.

⌋()

**[constr_3313] E2E transformer configuration** ⌈ For each TransformationDescription variant that is a EndToEndTransformationDescription

- attribute protocol of TransformationTechnology shall be set to |E2E|
- attribute version of TransformationTechnology shall be set to |1.0.0|
- attribute transformerClass of TransformationTechnology shall be set to |safety|

⌋()

**[constr_3314] BufferProperties . bufferComputation is mandatory** ⌈ The BufferProperties that is aggregated by TransformationTechnology in the role bufferProperties shall always define the bufferComputation .

⌋()

**[constr_3315] The value of V0 in BufferProperties . bufferComputation setting for a COM Based transformer** ⌈ The value of V0 of bufferComputation of a TransformationTechnology which has the protocol attribute set to COMBased shall have the same value as the length attribute of the ISignalIPdu to which the ISignalGroup is mapped. The ISignalGroup refers to the DataTransformation in the role comBasedSignalGroupTransformation which refers to a TransformationTechnology in the transformerChain .

⌋()

**[constr_3316] Allowed values for EndToEndTransformationDescription . maxDeltaCounter in PROFILE_07** ⌈ If the EndToEndTransformationDescription . profileName attribute has a value of PROFILE_07 the value of maxDeltaCounter attribute shall be in the range 1-4‘294‘967‘295.

⌋()

**[constr_3317] Assuring the same data interpretation on the sender and receiver sides in case of serialization based on the ImplementationDataType s** ⌈ In order to assure the same interpretation of the serialized data by the SOME/IP transformers on the sender and receiver sides in case of serialization based on either a primitive or a composite ImplementationDataType , the same SwBaseType shall be defined

- for this primitive DataPrototype or
- for each primitive DataPrototype of the leaf elements of the composite DataPrototype starting from the first element until and including the last element that is requested by the receiver,

by the ImplementationDataType s that either types the corresponding PortPrototype s on the top level Software Composition of the communicating EcuInstance s, or it is mapped to the ApplicationDataType that types it.

⌋()

**[constr_3318] Allowed use of ISignal . networkRepresentationProps** ⌈ If a reference from ISignal to DataTransformation in the role dataTransformation exists, this ISignal SHALL NOT aggregate SwDataDefProps in the role networkRepresentationProps .

⌋()

**[constr_3319] Existence of DataPrototypeTransformationProps . networkRepresentationProps** ⌈ ISignal . transformationISignalProps . dataPrototypeTransformationProps . networkRepresentationProps shall either

- not exist at all or
- shall be defined for all leaf elements of the root DataPrototype transmitted in the ISignal

⌋()

**[constr_3322] Consistent setting of SocketConnectionIpduIdentifier . pduCollectionSemantics in the context of one SocketConnectionBundle** ⌈ The value of the attribute SocketConnectionIpduIdentifier . pduCollectionSemantics shall be identical for all referenced SocketConnectionIpduIdentifier s within the context of a given SocketConnectionBundle .

⌋()

**[constr_3323] Relation between NmCluster . nmPncParticipation and PncMapping . pncGroup** ⌈ If a PncMapping references an ISignalIPduGroup in role pncGroup which in turn contains (either directly or via one of its subordinate ISignalIPduGroup s referenced in role containedISignalIPduGroup ) ISignalIPdu s that are referenced by a PduTriggering in role iPdu which in turn is composed by a PhysicalChannel in role pduTriggering which in turn is composed by CommunicationCluster in role physicalChannel which in turn is referenced by an NmCluster in role communicationCluster , then this NmCluster shall have its nmPncParticipation attribute set to TRUE.

⌋()

**[constr_3324] Category of SecureCommunicationFreshnessProps and SecureCommunicationAuthenticationProps** ⌈ SecureCommunicationFreshnessProps that is referenced by a SecuredIPdu in the role freshnessProps shall have the same category value as the SecureCommunicationAuthenticationProps that is referenced by the same SecuredIPdu in the role authenticationProps .

⌋()

**[constr_3325] `SecureCommunicationFreshnessProps` and `SecureCommunicationAuthenticationProps` attribute values for predefined categories** ⌈ Table `table_3a_SecurityProfiles` defines applicable attribute values for security profiles that are standardized by AUTOSAR.

⌋()

**[constr_3326] Allowed values for `EndToEndTransformationISignalProps . dataIdMode` in PROFILE_11** ⌈ If the `EndToEndTransformationDescription . profileName` attribute has a value of PROFILE_11 then the value of the `EndToEndTransformationDescription . dataIdMode` attribute shall be set to `all16Bit` or `lower12Bit` .

⌋()

**[constr_3327] Effect of `EndToEndTransformationDescription . upperHeaderBitsToShift` value in PROFILE_22** ⌈ If the `EndToEndTransformationDescription . profileName` attribute has a value of PROFILE_22, then `EndToEndTransformationDescription . offset` shall be set to the same value of `upperHeaderBitsToShift` .

⌋()

**[constr_3328] `SomeipTpConnection . transportPdu` reference restriction** ⌈ A `PduTriggering` that is referenced by a `SomeipTpConnection` in the role `transportPdu` shall reference a `GeneralPurposeIPdu` with category SOMEIP_SEGMENTED_IPDU in the role `iPdu` .

⌋()

**[constr_3329] `SomeipTpConnection . tpSdu` reference restriction** ⌈ A `PduTriggering` that is referenced by a `SomeipTpConnection` in the role `tpSdu` shall reference an `IPdu` in the role `iPdu` .

⌋()

**[constr_3330] Same `transportPdu` shall not be used in different `SomeipTpConnection` s** ⌈ A `PduTriggering` that is referencing a `GeneralPurposeIPdu` with category SOMEIP_SEGMENTED_IPDU in the role `iPdu` shall be referenced at most once by a `SomeipTpConnection` in the role `transportPdu` .

⌋()

**[constr_3331] Standardized values for the attribute `category` of meta-class `EthernetCommunicationConnector`** ⌈ The following values of the attribute `category` of meta-class `EthernetCommunicationConnector` are reserved by the AUTOSAR standard:

- WIRED: This represents the usage of the `EthernetCommunicationConnector` in case of a wired ethernet connection

- WIRELESS: This represents the usage of the EthernetCommunicationConnector in case of a wireless ethernet connection

⌋()

**[constr_3332] Standardized values for the attribute category of meta-class EthernetCommunicationController** ⌈ The following values of the attribute category of meta-class EthernetCommunicationController are reserved by the AUTOSAR standard:

- WIRED: This represents the usage of the EthernetCommunicationController in case of a wired ethernet connection
- WIRELESS: This represents the usage of the EthernetCommunicationController in case of a wireless ethernet connection

⌋()

**[constr_3333] Standardized values for the attribute category of meta-class EthernetPhysicalChannel** ⌈ The following values of the attribute category of meta-class EthernetPhysicalChannel are reserved by the AUTOSAR standard:

- WIRED: This represents the usage of the EthernetPhysicalChannel in case of a wired ethernet connection
- WIRELESS: This represents the usage of the EthernetPhysicalChannel in case of a wireless ethernet connection

⌋()

**[constr_3334] Allowed references between EthernetPhysicalChannel and EthernetCommunicationConnector** ⌈ An EthernetPhysicalChannel is only allowed to reference EthernetCommunicationConnector s in the role commConnector that have the same category value as the referencing EthernetPhysicalChannel .

⌋()

**[constr_3335] Allowed references between EthernetCommunicationConnector and EthernetCommunicationController** ⌈ An EthernetCommunicationConnector is only allowed to reference an EthernetCommunicationController in the role commController that has the same category value as the referencing EthernetCommunicationConnector .

⌋()

**[constr_3336] EthernetPhysicalChannel . soAdConfig in case of WIRELESS EthernetPhysicalChannel** ⌈ If EthernetPhysicalChannel has the category *WIRELESS* then the EthernetPhysicalChannel shall not aggregate the SoAdConfig .

⌋()

**[constr_3337] `IPduPort . useAuthDataFreshness` is configurable on the receiver side** ⌈ The `IPduPort . useAuthDataFreshness` attribute shall only be used in `IPduPort` s with the `communicationDirection` = in.

⌋()

**[constr_3338] `IPduPort . useAuthDataFreshness` validness** ⌈ The `IPduPort . useAuthDataFreshness` information is only valid for `SecuredIPdu` s.

⌋()

**[constr_3339] Relation between `authDataFreshnessStartPosition` , `authDataFreshnessLength` and `useAuthDataFreshness`** ⌈ If `authDataFreshnessStartPosition` and `authDataFreshnessLength` are set to a value for a `SecuredIPdu` then the `useAuthDataFreshness` shall be set as well to a value on all `IPduPort` s with `communicationDirection` = in that are referenced by a `PduTriggering` of the `SecuredIPdu` .

⌋()

**[constr_3364] `headerLength` shall be a multiple of 8** ⌈ The header length in bits specified by `headerLength` shall be a multiple of 8.

⌋()

**[constr_3365] `EthernetPhysicalChannel` s with different `category` values are not allowed within an `EthernetCluster`** ⌈ A mix of `EthernetPhysicalChannel` s with different `category` values within an `EthernetCluster` is currently not supported by AUTOSAR.

⌋()

**[constr_3373] Limitation on the number of `PhysicalChannel` s that are referencing a `CommunicationConnector`** ⌈ A `CommunicationConnector` shall only be referenced by at most one `PhysicalChannel` .

⌋()

**[constr_3378] Maximal one `AliasNameAssignment` allowed per `FlatInstanceDescriptor`** ⌈ In a given instance of `AliasNameSet` in the bound system there shall be at most one `aliasName` per `FlatInstanceDescriptor` .

⌋()

**[constr_3379] Multiple `SocketAddress` entries with the same IP Address, Protocol and Port in the context of a given `EcuInstance`** ⌈ If there are two or more `SocketAddress` entities within the scope of one `SoAdConfig` in the scope of one `EcuInstance` that have the same static (fixed at configuration time) IP Address, Protocol and Port in the aggregated `ApplicationEndpoint` and `NetworkEndpoint` , (e.g., 192.168.1.1, Tcp and 10000, respectively), `ProvidedServiceInstance` / `ConsumedServiceInstance` may only be defined in the `ApplicationEndpoint` aggregated by one of these `SocketAddress` entries.

⌋()

**[constr_3383] Standardized values for the attribute `category` of meta-class `GeneralPurposeConnection`** ⌈ The following values of the attribute `category` of meta-class `GeneralPurposeConnection` are reserved by the AUTOSAR standard:

- XcpChannel

⌋()

**[constr_3384] `PduTriggering` s referenced by `GeneralPurposeConnection` shall be defined on the same `PhysicalChannel`** ⌈ The `PduTriggering` s that are referenced by the `GeneralPurposeConnection` in the role `pduTriggering` shall be defined on the same `PhysicalChannel` .

⌋()

**[constr_3385] XcpChannel is allowed to reference exactly two `PduTriggering` s** ⌈ In case that the `category` of meta-class `GeneralPurposeConnection` is set to the value *XcpChannel* the `GeneralPurposeConnection` is allowed to reference exactly two `PduTriggering` s in the role `pduTriggering` .

⌋()

**[constr_3386] XcpChannel is only allowed to reference `PduTriggering` s of `GeneralPurposeIPdu` s with category XCP** ⌈ In case that the `category` of meta-class `GeneralPurposeConnection` is set to the value *XcpChannel* the `GeneralPurposeConnection` is allowed to reference `PduTriggering` s of `GeneralPurposeIPdu` s with category XCP.

⌋()

**[constr_3399] Existence of `securedAreaOffset` and `securedAreaLength`** ⌈ If the `securedAreaOffset` is defined then the `securedAreaLength` shall be defined as well and vice versa.

⌋()

**[constr_3400] Usage of `SdClientConfig` attributes in `ConsumedServiceInstance` and `ConsumedEventGroup`** ⌈ Usage of `SdClientConfig` attributes in `ConsumedServiceInstance` and `ConsumedEventGroup` shall follow the restrictions given in `table_3a_SdClientAttributes` .

⌋()

**[constr_3401] Usage of `SdServerConfig` attributes in `ProvidedServiceInstance` and `EventHandler`** ⌈ Usage of `SdServerConfig` attributes in `ProvidedServiceInstance` and `EventHandler` shall follow the restrictions given in `table_3a_SdServerAttributes` .

⌋()

**[constr_3402] Mandatory `offset` if `noHeader` is used** ⌈ For each `IPdu` which is assigned to a `ContainerIPdu` in the role `containedPduTriggering` with `ContainerIPdu.headerType` = `noHeader` the `IPdu.containedIPduProps.offset` shall be defined.

⌋()

**[constr_3403] Usage of `ContainerIPdu.rxAcceptContainedIPdu` if `noHeader` is used** ⌈ If the `ContainerIPdu.headerType` is set to `noHeader` then the `ContainerIPdu.rxAcceptContainedIPdu` attribute value shall be set to `acceptConfigured`.

⌋()

**[constr_3404] Usage of `ContainedIPduProps.updateIndicationBitPosition`** ⌈ `ContainedIPduProps.updateIndicationBitPosition` is only allowed to be set to a value if the `headerType` of the `ContainerIPdu` that contains the `IPdu` with `containedIPduProps` is set to `noHeader`.

⌋()

**[constr_3405] Dynamic Length `IPdu` inside of a static configured `ContainerIPdu`** ⌈ Only the last contained `IPdu` (according to the `ContainedIPduProps.offset` ) of a `ContainerIPdu` with static container layout (i.e., a `ContainerIPdu` with `headerType` set to `noHeader` ) is allowed to be a dynamic length `IPdu` (i.e, a contained `IPdu` that at runtime may exhibit a length different from the one statically configured via `Pdu.length` of the respective `Pdu` ). All other contained `IPdu` s of a `ContainerIPdu` with static container layout have to be static length `IPdu` s.

⌋()

**[constr_3406] All signals before `authDataFreshnessStartPosition` shall have a static length** ⌈ In case that

- an `ISignalIPdu` is referenced by the `SecuredIPdu` with the `payload` reference via the `PduTriggering` and
- the `authDataFreshnessStartPosition` and `authDataFreshnessLength` define the area in the `ISignalIPdu` that is taken to verify and generate the Freshness then

all `ISignal` s that are mapped into the `ISignalIPdu` in front of the configured `authDataFreshnessStartPosition` shall have a static length.

⌋()

**[constr_3407] Freshness Value in Authentic `IPdu` is not allowed to be used in case of `ContainerIPdu` with a dynamic layout** ⌈ If a `ContainerIPdu` that is referenced by the `SecuredIPdu` with the `payload` reference via the `PduTriggering` contains a dynamic layout (i.e. `ContainerIPdu.headerType` is set to `longHeader` or `shortHeader` ) and multiple contained `IPdu` s then each `IPduPort` that is refer-

enced by the PduTriggering of the SecuredIPdu shall have the attribute useAuthDataFreshness set to false.

⌋()

**[constr_3501] Role of SystemSignal in 1:n communication** ⌈ In case of 1:n communication the VariableDataPrototype in the PPortPrototype of the SwComponentPrototype shall be mapped to only one SystemSignal .

⌋()

**[constr_3506] Mapping of composite data type to SystemSignal s in SystemSignalGroup** ⌈ The elements of a composite data type shall be mapped to single SystemSignal s which shall be members of one SystemSignalGroup if no data transformation (except COM Based Transformer) is used.

There are two exceptions to this rule:

- it is allowed to map an array VariableDataPrototype consisting of UINT8 elements to exactly one SystemSignal in the context of one SenderReceiverToSignalMapping (see section sec_3a_Mapping_of_20_Data_Elements_with_primitive_datatypes_on_System ).
- in case the COM Based Transformer [13] is used it is the integral part of the approach to have a fixed mapping of the individual elements of composite data types to SystemSignal s in a SystemSignalGroup ( TPS_SYST_02058 ).

⌋()

**[constr_3508] Value of nmReadySleepTime** ⌈ The nmReadySleepTime value shall be a multiple of cycle * nmRepetitionCycle .

⌋()

**[constr_3514] No two ISignalToIPduMapping s shall reference the identical ISignal** ⌈ No two ISignalToIPduMapping s shall reference the identical ISignal in the role iSignal in the scope of one System.

⌋()

**[constr_3515] Fully filled EthernetPriorityRegeneration table** ⌈ In case the CouplingPortDetails . ethernetPriorityRegeneration is defined it shall contain exactly 8 elements of EthernetPriorityRegeneration , one for each value of ingressPriority (0-7).

⌋()

**[constr_3516] limitation of Pdu . length for CAN L-PDUs** ⌈ The Pdu . length of CAN PDUs shall be restricted to 0..8 for classic CAN L-PDUs and 0..8, 12, 16, 20, 24, 32, 48, 64 for CAN FD L-PDUs.

⌋()

**[constr_3517] Consistent setting of `ContainedIPduProps . collectionSemantics` in the context of one `ContainerIPdu`** ⌈ The value of the attribute `ContainedIPduProps . collectionSemantics` shall be identical for all contained `IPdu` s within the context of a given `ContainerIPdu` .

⌋()

**[constr_3518] Range of CanControllerFdConfiguration.paddingValue and CanControllerFdConfigurationRequirements.paddingValue** ⌈ The value given for `CanControllerFdConfiguration . paddingValue` and `CanControllerFdConfigurationRequirements . paddingValue` shall be in the range from 0 to 255.

⌋()

**[constr_3519] Value of `category` of `GlobalTimeDomain`** ⌈ The attribute `category` of `GlobalTimeDomain` can have the following values:

- SYNCHRONIZED: this time base does not depend on the existence of another time base
- OFFSET: this time base depends on the existence of another time base. It delivers a value that represents an offset relative to the referenced ( `GlobalTimeDomain . offsetTimeDomain` ) synchronized time base.

⌋()

**[constr_3520] Offset time domain shall be based on a synchronized time domain** ⌈ If a `GlobalTimeDomain` has a reference with the role `GlobalTimeDomain . offsetTimeDomain` the reference source shall have a `GlobalTimeDomain . domainId` in the range of 16-31 and the reference target shall have a `GlobalTimeDomain . domainId` in the range of 0-15.

⌋()

**[constr_3521] `defaultVlan` and `vlanMembership`** ⌈ If a `CouplingPort` refers to an `EthernetPhysicalChannel` in the role `defaultVlan` the `CouplingPort` shall also have a `vlanMembership` defined. This `VlanMembership` shall point to the same `EthernetPhysicalChannel` in the role `vlan` as the `defaultVlan` .

⌋()

**[constr_3522] `vlanModifier` and `vlanMembership`** ⌈ If a `CouplingPort` refers to an `EthernetPhysicalChannel` in the role `vlanModifier` the `CouplingPort` shall also have a `vlanMembership` defined. This `VlanMembership` shall point to the same `EthernetPhysicalChannel` in the role `vlan` as the `vlanModifier` .

⌋()

**[constr_3523] `CouplingPort` and `PncMapping` in the scope of an `EthernetPhysicalChannel`** ⌈ If

- a `CouplingPort` referring to an `EthernetPhysicalChannel` – via a `VlanMembership` – references at least one `PncMapping`

- and that PncMapping contains PDUs – via the assignment of PncMapping . pncGroup – that are transported on this EthernetPhysicalChannel

then every CouplingPort referring to that EthernetPhysicalChannel shall reference at least one PncMapping as well.

⌋()

**[constr_3524] Definition of couplingPortRole on CouplingPort for managed CouplingElement** ⌈ A managed CouplingElement shall have either

- at most one CouplingPort with couplingPortRole set to hostPort or
- at least one CouplingPort with couplingPortRole set to upLinkPort .

⌋()

**[constr_3525] Connection of CouplingPort with couplingPortRole set to upLinkPort** ⌈ A CouplingPort with couplingPortRole set to upLinkPort shall be connected to exactly one other CouplingPort with couplingPortRole set to upLinkPort .

⌋()

## 2.20 TPS_TimingExtensions

**[constr_4500] Restricted usage of functions** ⌈ The functions *TIMEX_occurs* , *TIMEX_hasOccurred* , *TIMEX_timeSinceLastOccurrence* , *TIMEX_angleSinceLast Occurrence* , and *TIMEX_modeActive* can only be used for occurrence expressions, which are applied to events of type TDEventComplex .

⌋()

**[constr_4501] Application rule for the occurrence expression in TDEventComplex** ⌈ The occurrence expression shall be specified such that it describes an *event* rather than a state. As a consequence the occurrence expression must ensure that a complex timing event *could* only occur at the occurrence time of one of the referenced TimingDescriptionEvent s.

⌋()

**[constr_4502] Use references only as function operands** ⌈ The references to model elements (e.g. the *timing event* reference targeting TimingDescriptionEvent ) do have specific semantics. The usage of these references within the expression is *only* allowed as operand of the functions mentioned above.

⌋()

**[constr_4503] Restricted usage of AutosarOperationArgumentInstance for Content Filter** ⌈ If a content filter is defined for an atomic event then references to AutosarOperationArgumentInstance s are only allowed if the atomic event is of

type TDEventOperation . Only if such an atomic event occurs, the value of the operation arguments can be evaluated. Thus, also the scope of the atomic event must be the same as the AutosarOperationArgumentInstance , meaning that they must point to the same ClientServerOperation . Finally, references to an AutosarOperationArgumentInstance with argument direction ”out” are only allowed, if the atomic event of type TDEventOperation refers either to the point in time when the operation call response has been sent (TD-EVENT-OPERATION-TYPE=OPERATION-CALL-RESPONSE-SENT) or to the point in time when the operation call response has been received (TD-EVENT-OPERATION-TYPE=OPERATION-CALL-RESPONSE-RECEIVED).

⌋()

**[constr_4504] Restricted usage of AgeConstraint** ⌈ An AgeConstraint shall only be defined for events of type TimingDescriptionEvent associated with the receipt and reading of data.

⌋()

**[constr_4505] Specifying minimum and maximum number of occurrences** ⌈ The minimum and maximum number of occurrences shall be specified such that the following holds: $0 \leq$ minNumberOfOccurrences $\leq$ maxNumberOfOccurrences .

⌋()

**[constr_4506] Specifying minimum inter-arrival time and pattern length** ⌈ The minimum inter-arrival time and pattern length shall be specified such that the following holds: $0 <$ minimumInterArrivalTime $\leq$ patternLength .

⌋()

**[constr_4507] Specifying pattern length, pattern jitter and patter period** ⌈ The pattern length, pattern jitter and pattern period shall be specified such that the following holds: patternLength + patternJitter $<$ patternPeriod .

⌋()

**[constr_4508] TDEventVfb shall reference PortPrototypeBlueprint only in Blueprints** ⌈ An event type TDEventVfb only shall reference PortPrototype-Blueprint in blueprints.

⌋()

**[constr_4509] Only VfbTiming shall be a Blueprint** ⌈ Only the VfbTiming is blueprintable.

⌋()

**[constr_4510] Specifying references to RunnableEntity and VariableAccess** ⌈ A RunnableEntity and VariableAccess shall be referenced at the same time if and only if the value of tdEventSwcInternalBehaviorType is "runnableEntity VariableAccess". These two references are not mutual exclusive.

⌋()

**[constr_4511] Validity of referencing `RunnableEntity`** ⌈ A `RunnableEntity` shall be referenced if and only if the value of `tdEventSwcInternalBehaviorType` is "runnableEntityActivated", "runnableEntityStarted", "runnableEntityTerminated", or "runnableEntityVariableAccess".

⌋()

**[constr_4512] Validity of referencing `VariableAccess`** ⌈ A `VariableAccess` shall be referenced if and only if the value of `tdEventSwcInternalBehaviorType` is "runnableEntityVariableAccess".

⌋()

**[constr_4513] `SynchronizationTimingConstraint` shall reference at least two events** ⌈ In the case, that the `SynchronizationTimingConstraint` is imposed on events then at least two (2) timing description events shall be referenced.

⌋()

**[constr_4514] `SynchronizationTimingConstraint` shall reference at least two event chains** ⌈ In the case, that the `SynchronizationTimingConstraint` is imposed on event chains then at least two (2) timing description event chains shall be referenced.

⌋()

**[constr_4515] Specifying stimulus and response in `TimingDescriptionEventChain`** ⌈ The references between `TimingDescriptionEventChain` and `TimingDescriptionEvent` playing the role `stimulus` and `response` shall not reference the same `TimingDescriptionEvent` .

⌋()

**[constr_4516] Specifying event chain segments** ⌈ If a `TimingDescriptionEventChain` consists of further event chain segments then at least one sequence of event chain segments shall exists from the event chain's `stimulus` to the `response` .

⌋()

**[constr_4517] Referencing no further event chain segments** ⌈ If a `TimingDescriptionEventChain` is not subdivided in further event chain segments, then the reference playing the role of `segment` shall reference this `TimingDescriptionEventChain` . In other words, an event chain without any event chain segment shall reference itself.

⌋()

**[constr_4518] Specifying `stimulus` event and `response` event of first and last event chain segment** ⌈ The `stimulus` event of the first event chain segment and the

response event of the last event chain segment shall reference the stimulus and response of the parent event chain the event chain segments directly belong to.

⌋()

**[constr_4519] Specifying patternLength** ⌈ The patternLength shall be specified such that the following holds: $0 \leq max(\texttt{offset}) \leq \texttt{patternLength}$ .

⌋()

**[constr_4520] Specifying attribute synchronizationConstraintType** ⌈ The attribute synchronizationConstraintType shall be specified if the SynchronizationTimingConstraint is imposed on events.

⌋()

**[constr_4521] Specifying attribute synchronizationConstraintType** ⌈ The attribute synchronizationConstraintType shall be specified if the SynchronizationTimingConstraint is imposed on event chains.

⌋()

**[constr_4522] SynchronizationTimingConstraint shall either reference events or event chains** ⌈ The SynchronizationTimingConstraint shall either reference timing description events or timing description event chains, but not both at the same time.

⌋()

**[constr_4523] Specifying attributes maxCycles and maxSlots** ⌈ The optional attributes maxCycles and maxSlots shall never be specified in any element EOCExecutableEntityRefGroup that is part of a hierarchical execution order constraint.

⌋()

**[constr_4524] Referencing TimingDescriptionEvent** ⌈ Any element EOCExecutableEntityRefGroup that is part of a hierarchical execution order constraint shall not reference any timing description event TimingDescriptionEvent .

⌋()

**[constr_4525] Precedence of successor relationships successor and directSuccessor** ⌈ The successor relationships successor and directSuccessor take always precedence over the ordered multiplicity of the association nestedElement .

⌋()

**[constr_4526] Specifying maxCycles and maxSlots in a Repetitive Execution Order Constraint** ⌈ The optional attributes maxCycles and maxSlots shall be specified only by the *root* group of executable entity references EOCExecutableEntityRefGroup .

⌋()

**[constr_4527] Referencing `TimingDescriptionEvent` in a Repetitive Execution Order Constraint** ⌈ The `TimingDescriptionEvent` shall be specified only by the *root* group of executable entity references `EOCExecutableEntityRefGroup` .

⌋()

**[constr_4528] The *root* `EOCExecutableEntityRefGroup` shall reference only `EOCExecutableEntityRefGroup` s** ⌈ The *root* `EOCExecutableEntityRefGroup` shall reference only groups of executable entity references respectively event references grouped by the element `EOCExecutableEntityRefGroup` s.

⌋()

**[constr_4529] Number of nested elements referenced by the *root* `EOCExecutableEntityRefGroup`** ⌈ The number of nested elements referenced by the *root* `EOCExecutableEntityRefGroup` shall be exactly the number given by the attribute `maxCycles` .

⌋()

**[constr_4530] An `EOCExecutableEntityRefGroup` representing a cycle shall reference only `EOCExecutableEntityRef` s respectively `EOCEventRef` s** ⌈ The `EOCExecutableEntityRefGroup` representing a cycle shall reference only executable entity references `EOCExecutableEntityRef` s respectively event references `EOCEventRef` s.

⌋()

**[constr_4531] Number of nested elements referenced by `EOCExecutableEntityRefGroup` representing a cycle** ⌈ The number of nested elements referenced by a `EOCExecutableEntityRefGroup` representing a cycle shall be exactly the number given by the attribute `maxSlots` .

⌋()

**[constr_4532] Successor relationship is not self-referencing** ⌈ The target and source of the successor relationships `successor` and `directSuccessor` shall not be the same. In other words an `EOCExecutableEntityRef` and `EOCExecutableEntityRefGroup` shall not reference itself as its logical or direct successor.

⌋()

**[constr_4533] Maximum number of successor relationships** ⌈ The maximum number of successor relationships, namely `successor` or `directSuccessor` , between two `EOCExecutableEntityRef` s, between two `EOCEventRef` s, between two `EOCExecutableEntityRefGroup` s, between an `EOCExecutableEntityRef` and an `EOCExecutableEntityRefGroup` , or between an `EOCEventRef` and an `EOCExecutableEntityRefGroup` is one (1).

⌋()

**[constr_4534] Maximum number of `directSuccessor` relationships** ⌈ The number of `directSuccessor` relationships of an `EOCExecutableEntityRef` , an `EOCEventRef` , or an `EOCExecutableEntityRefGroup` shall not exceed the number of independent execution units available in a system.

⌋()

**[constr_4535] An `ExecutionOrderConstraint` needs to be consistent regarding effective modes** ⌈ In case of an `ExecutionOrderConstraint` using events there exists a mode in which all referenced events are enabled; in other words the events are *not* disabled. In case of an `ExecutionOrderConstraint` using `ExecutableEntity` s there exists a mode in which all referenced `ExecutableEntity` s are enabled and `ExecutableEntity` s without any event are considered to be always enabled. If `ExecutableEntity` s are started by a single event then this particular event is considered and for `ExecutableEntity` s with multiple events the superset of the related modes is considered.

⌋()

**[constr_4536] Compatible recurrence of any `ExecutableEntity`** ⌈ In an `ExecutionOrderConstraint` the `ExecutableEntity` s, referenced by all `EOCExecutableEntityRef` s respectively all `EOCEventRef` s, shall be compatible with regard to their recurrence.

⌋()

**[constr_4537] References among elements in an `ExecutionOrderConstraint`** ⌈ An `EOCExecutableEntityRef` respectively `EOCEventRef` or an `EOCExecutableEntityRefGroup` shall reference only `EOCExecutableEntityRef` s, respectively all `EOCEventRef` s, or `EOCExecutableEntityRefGroup` s which are part of the same `ExecutionOrderConstraint` .

⌋()

**[constr_4538] Hierarchical Execution Order Constraint: `EOCExecutableEntityRef` , `EOCEventRef` , and `EOCExecutableEntityRefGroup` shall be target or source of a successor relationship** ⌈ In a given Hierarchical Execution Order Constraint, each `EOCExecutableEntityRef` , `EOCEventRef` , and `EOCExecutableEntityRefGroup` which is not part of an `EOCExecutableEntityRefGroup` shall be target or source of at least one successor relationship.

⌋()

**[constr_4539] The successor relationships `successor` and `directSuccessor` shall not be used** ⌈ The successor relationships `successor` and `directSuccessor` shall not be used in a Repetitive Execution Order Constraint.

⌋()

**[constr_4540] `maxCycles` and `maxSlots` shall not be zero** ⌈ If the optional attributes `maxCycles` and `maxSlots` are used, then the values of the optional attributes `maxCycles` and `maxSlots` shall be greater than zero (0).

⌋()

**[constr_4541] `EOCExecutableEntityRef` shall reference `ExecutableEntity` in Ordinary Execution Order Constraint** ⌈ In an Ordinary Execution Order Constraint all `EOCExecutableEntityRef` s shall reference an `ExecutableEntity` .

⌋()

**[constr_4542] `EOCExecutableEntityRef` shall reference `ExecutableEntity` in Hierarchical Execution Order Constraint** ⌈ In an Hierarchical Execution Order Constraint all `EOCExecutableEntityRef` s shall reference an `ExecutableEntity` .

⌋()

**[constr_4543] Maximum value of the parameter `minimumInterArrivalTime`** ⌈ The value of the parameter `minimumInterArrivalTime` shall be less than or equal the value of the parameter `period` .

⌋()

**[constr_4544] Specifying `patternLength` , `patternJitter` and `patternPeriod`** ⌈ The pattern length, pattern jitter and pattern period shall be specified such that the following holds: `patternLength` + `patternJitter` < `patternPeriod` .

⌋()

**[constr_4545] Referring either `ExecutableEntity` s or `AbstractEvent` s** ⌈ An `ExecutionOrderConstraint` shall contain either only `EOCExecutableEntityRef` or only `EOCEventRef` , but not both. In the former case `ExecutableEntity` s are referenced and in the latter case `AbstractEvent` s are referenced.

⌋()

**[constr_4546] Setting the attribute `isEvent`** ⌈ The value of the attribute `isEvent` shall be set to ”TRUE” if and only if the execution order constraint refers to events only (refer to constr_4545 ). The value of the attribute `isEvent` shall be set to ”FALSE” if and only if the execution order constraint refers to executable entities only (refer to constr_4545 ).

⌋()

**[constr_4547] Setting the attribute `permitMultipleReferencesToEE`** ⌈ The value of the attribute `permitMultipleReferencesToEE` shall be specified if and only if the value of the attribute `isEvent` (refer to constr_4546 ) is set to ”FALSE”. In other words specifying whether an executale entity is permitted to be referenced more than once in an execution order constraint is only allowed in case of an execution order constraint refering to executable entities only.

⌋()

**[constr_4548] `EOCEventRef` shall reference `AbstractEvent` in Ordinary Execution Order Constraint** ⌈ In an Ordinary Execution Order Constraint all `EOCEventRef` s shall reference an `AbstractEvent` .

⌋()

**[constr_4549] `EOCEventRef` shall reference `AbstractEvent` in Hierarchical Execution Order Constraint** ⌈ In an Hierarchical Execution Order Constraint all `EOCEventRef` s shall reference an `AbstractEvent` .

⌋()

**[constr_4550] A Hierarchical Execution Order Constraint shall have an unambiguous root `EOCExecutableEntityRefGroup`** ⌈ A Hierarchical Execution Order Constraint may contain multiple `orderedElement` s, which may be any combination of any number of `EOCExecutableEntityRef` s respectively `EOCEventRef` s and `EOCExecutableEntityRefGroup` s. Among these needs to be exactly one `EOCExecutableEntityRefGroup` being neither target nor source of any `successor` or `directSuccessor` relationship. This `EOCExecutableEntityRefGroup` is the *root* of the Hierarchical Execution Order Constraint.

⌋()

**[constr_4551] Use only Numericals in `TDEventOccurrenceExpression`** ⌈ The target data prototype of the instance references of `variable` and `argument` shall be `Numerical` .

⌋()

**[constr_4552] Restricted usage of `AutosarVariableInstance` for Content Filter** ⌈ If a content filter is defined for an atomic event then references to `AutosarVariableInstance` s are only allowed if the atomic event is of type `TDEventVariableDataPrototype` . Only if such an atomic event occurs, the value of the variables can be evaluated. Thus, also the scope of the atomic event must be the same as the `AutosarVariableInstance` , meaning that they must point to the same `VariableDataPrototype` .

⌋()

## 2.21 TR_FrancaIntegration

**[TR_FRANCA_CONSTR_00010] Franca connector has no duplicate links** ⌈ There must not be two links with the same AUTOSAR and Franca sides in a Franca connector.

⌋()

**[TR_FRANCA_CONSTR_00020] Franca connector has no client server fan out** ⌈ A required client server port of an AUTOSAR component prototype must not be connected to more than one Franca instance.

⌋()