Document Title	TPS_SystemTemplate: Complete Change Documentation 4.3.0 - 4.3.1
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	695

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Table of Contents

1 TPS_SystemTemplate 4

1.1 Specification Item constr_2025 . . . 4
1.2 Specification Item constr_3052 . . . 5
1.3 Specification Item constr_3053 . . . 7
1.4 Specification Item constr_3136 . . . 8
1.5 Specification Item constr_3139 . . . 9
1.6 Specification Item constr_3364 . . . 17
1.7 Specification Item constr_3365 . . . 19
1.8 Specification Item constr_3373 . . . 20
1.9 Specification Item constr_3378 . . . 21
1.10 Specification Item constr_3379 . . . 22
1.11 Specification Item constr_3383 . . . 23
1.12 Specification Item constr_3384 . . . 25
1.13 Specification Item constr_3385 . . . 27
1.14 Specification Item constr_3386 . . . 28
1.15 Specification Item constr_3399 . . . 30
1.16 Specification Item constr_3400 . . . 31
1.17 Specification Item constr_3401 . . . 34
1.18 Specification Item constr_3402 . . . 38
1.19 Specification Item constr_3403 . . . 41
1.20 Specification Item constr_3404 . . . 44
1.21 Specification Item constr_3405 . . . 47
1.22 Specification Item constr_3406 . . . 50
1.23 Specification Item constr_3407 . . . 59
1.24 Specification Item TPS_SYST_01120 . . . 67
1.25 Specification Item TPS_SYST_02005 . . . 68
1.26 Specification Item TPS_SYST_02098 . . . 69
1.27 Specification Item TPS_SYST_02100 . . . 72
1.28 Specification Item TPS_SYST_02112 . . . 75
1.29 Specification Item TPS_SYST_02160 . . . 77
1.30 Specification Item TPS_SYST_02162 . . . 77
1.31 Specification Item TPS_SYST_02163 . . . 79
1.32 Specification Item TPS_SYST_02164 . . . 80
1.33 Specification Item TPS_SYST_02165 . . . 81
1.34 Specification Item TPS_SYST_02166 . . . 84
1.35 Specification Item TPS_SYST_02167 . . . 87
1.36 Specification Item TPS_SYST_02168 . . . 90
1.37 Specification Item TPS_SYST_02169 . . . 92
1.38 Specification Item TPS_SYST_02170 . . . 93
1.39 Specification Item TPS_SYST_02171 . . . 94

1.40 Specification Item TPS_SYST_02172 95
1.41 Specification Item TPS_SYST_02173 97
1.42 Specification Item TPS_SYST_02174 99
1.43 Specification Item TPS_SYST_02175 102
1.44 Specification Item TPS_SYST_02176 106
1.45 Specification Item TPS_SYST_02177 109
1.46 Specification Item TPS_SYST_02178 112
1.47 Specification Item TPS_SYST_02179 116
1.48 Specification Item TPS_SYST_02180 119
1.49 Specification Item TPS_SYST_02181 122
1.50 Specification Item TPS_SYST_02182 126
1.51 Specification Item TPS_SYST_02183 129
1.52 Specification Item TPS_SYST_02184 132
1.53 Specification Item TPS_SYST_02185 136
1.54 Specification Item TPS_SYST_02186 139
1.55 Specification Item TPS_SYST_02187 142
1.56 Specification Item TPS_SYST_02188 145
1.57 Specification Item TPS_SYST_02189 149

1 TPS_SystemTemplate

1.1 Specification Item constr_2025

Trace References:

none

Content:

In the With the exception of RunnableEntities that are subject to 1234 (RunnableEntities owned by NvBlockSwComponentTypes), in the context of a single EcuInstance , the values of the RunnableEntity.RunnableEntity.symbol in combination with the attribute Atomic SwComponentType.ImplementationProps.symbol of the meta-class SymbolProps owned by AtomicSwComponentType .symbol of all deployed RunnableEntitys Entities shall be unique such that no two (or more) combinations of RunnableEntity.RunnableEntity.symbol and the ImplementationProps.symbol of the meta-class SymbolProps owned by Atomic SwComponentType .in the role AtomicSwComponentType.symbolProps share the same value.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75258: [constr_2025] is to strict to support [constr_1234]

  Problem description:

  [constr_2025] Uniqueness of symbol attributes d In the context of a single EcuInstance,
  the values of the RunnableEntity.symbol in combination with the attribute
  AtomicSwComponentType.symbol of all deployed RunnableEntitys shall
  be unique such that no two (or more) combinations of RunnableEntity.symbol and
  AtomicSwComponentType.symbol share the same value. c()

  BUT

  [constr_1234] Value of RunnableEntity.symbol d 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. c()

  With the given rational:

  For example, RunnableEntity.symbol owned by an NvBlockSwComponentType

could rightfully be set to NvM_ReadBlock [33] but an arbitrary value like ReadThis Block is not permitted.
The rationale for [constr_1234] is that the RunnableEntitys that are triggered by an OperationInvokedEvent are not existing as such but are mapped to the respective function calls of the NvM. For more details of how this mapping can be achieved please
refer to [7].

Requires exactly the identical RunnableEntity.symbol is used by many NvBlockSwComponentType. Since for those the NvBlockSwComponentType.symbolProps are meaningless for RTE and therefor usually left empty.

By the way AtomicSwComponentType.symbolProps should be used instead AtomicSwComponentType.symbol

Agreed solution:

Rephrase constr_2025:

[constr_2025] Uniqueness of symbol attributes d With the exception of RunnableEntities that are subject to constr_1234 (RunnableEntities owned by NvBlockSwComponentTypes), in the context of a single EcuInstance the values of the RunnableEntity.symbol in combination with the attribute AtomicSwComponentType.symbol of all deployed RunnableEntitys shall
be unique such that no two (or more) combinations of RunnableEntity.symbol and AtomicSwComponentType.symbol share the same value. c()
–Last change on issue 75258 comment 4–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.2 Specification Item constr_3052

Trace References:

none

Content:

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.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74600: Signal Mapping by shortName

  Problem description:

  There's a not in the document that breaks:

  "By default, identical shortNames of ISignal elements identify correlating ISignals within the respective ISignalGroups referenced in the scope of ISignalMapping."

  I'm not sure what the background for this note is and whether it should be made a specification item (especially as the shortName-based routing is mentioned in TPS_Syst_01120.

  On the other hand: the model allows for a dedicated mapping of routing relations. Why is it necessary to have a name-based, implicit routing relation?

  Agreed solution:

  Change the note behind constr_3052 to a new specification item:

  [TPS_SYST_xxxx1] Routing of ISignals of ISignalGroups
  When performing a signal group routing two approaches are supported for the pairing of the included ISignals:
  - implicit mapping: the ISignalMapping points in the sourceSignal role to an ISignalTriggering of an ISignalGroup and no ISignalMappings are defined for the included ISignals. Identical shortNames of ISignal elements identify correlating ISignals between the source and the target in the scope of the ISignalMapping.
  - explicit mapping: the ISignalMapping points in the sourceSignal role to an ISignalTriggering of an ISignalGroup and in addition explicitly specified ISignalMappings define which ISignals correlate to each other.

  Change [TPS_SYST_01120] to (add implicit mapping):
  [TPS_SYST_01120] Precedence of ISignalMappings
  If a dedicated ISignalMapping for at least one ISignal within an ISignalGroup exists the implicit mapping on the basis of shortNames is no longer applicable for any ISignal within that ISignalGroup.

  Reference in [constr_3053] to the new spec item:
  [constr_3053] Complete ISignalMapping of target ISignalGroup
  If an ISignalGroup is referenced by a targetSignal then [TPS_SYST_01120] applies for each of the contained ISignal of that ISignalGroup.
  –Last change on issue 74600 comment 8–

BW-C-Level:

Application	Specification	Bus
1	3	1

1.3 Specification Item constr_3053

Trace References:

none

Content:

If an ISignalGroup is referenced by a ISignalMapping.targetSignal there shall exist either an explicit or an implicit mapping (see then TPS_SYST_01120 for each 02162 applies for each of the contained ISignal of that ISignalGroup.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74600: Signal Mapping by shortName

  Problem description:

  There's a not in the document that breaks:

  "By default, identical shortNames of ISignal elements identify correlating ISignals within the respective ISignalGroups referenced in the scope of ISignalMapping."

  I'm not sure what the background for this note is and whether it should be made a specification item (especially as the shortName-based routing is mentioned in TPS_Syst_01120.

  On the other hand: the model allows for a dedicated mapping of routing relations. Why is it necessary to have a name-based, implicit routing relation?

  Agreed solution:

  Change the note behind constr_3052 to a new specification item:

  [TPS_SYST_xxxx1] Routing of ISignals of ISignalGroups
  When performing a signal group routing two approaches are supported for the pairing of the included ISignals:
  - implicit mapping: the ISignalMapping points in the sourceSignal role to an ISignalTriggering of an ISignalGroup and no ISignalMappings are defined for the included ISignals. Identical shortNames of ISignal elements identify correlating ISignals

between the source and the target in the scope of the ISignalMapping.
- explicit mapping: the ISignalMapping points in the sourceSignal role to an ISignal-Triggering of an ISignalGroup and in addition explicitly specified ISignalMappings define which ISignals correlate to each other.

Change [TPS_SYST_01120] to (add implicit mapping):
[TPS_SYST_01120] Precedence of ISignalMappings
If a dedicated ISignalMapping for at least one ISignal within an ISignalGroup exists the implicit mapping on the basis of shortNames is no longer applicable for any ISignal within that ISignalGroup.

Reference in [constr_3053] to the new spec item:
[constr_3053] Complete ISignalMapping of target ISignalGroup
If an ISignalGroup is referenced by a targetSignal then [TPS_SYST_01120] applies for each of the contained ISignal of that ISignalGroup.
–Last change on issue 74600 comment 8–

BW-C-Level:

Application	Specification	Bus
1	3	1

1.4 Specification Item constr_3136

Trace References:

none

Content:

SecuredIPdus are allowed to reference PduTriggerings of ISignalIPdus, Container IPdus, DcmIPdus, MultiplexedIPdus, GeneralPurposeIPdus with Identifiable.category SOMEIP_SEGMENTED_IPDU and UserDefinedIPdus.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77428: Allow to secure GeneralPurposeIPdus

  Problem description:

  constr_3136 does not allow GeneralPurposeIPdus as payload of SecuredIPdus. Since use cases exist to secure such Pdus the list in constr_3136 shall be extended.

  Agreed solution:

Add GeneralPurposeIPdu to the list in constr_3136:

[constr_3136] Allowed payload of SecuredIPdus
SecuredIPdus are allowed to reference PduTriggerings of ISignalIPdus, ContainerIPdus, DcmIPdus, MultiplexedIPdus, GeneralPurposeIPdus with category SOMEIP_SEGMENTED_IPDU and UserDefinedIPdus.
–Last change on issue 77428 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.5 Specification Item constr_3139

Trace References:

none

Content:

The IPduPort.IPduPort.rxSecurityVerification is allowed to be set to false only for Secured IPdus with a static and fixed payload layout. For SecuredIPdus that contain dynamic length IPdus this attribute shall be always set to true.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77336: [SECOC] Dynamic length PDUs (Container) not possible / clear

  Problem description:

  The SecOC SWS does not make any assumptions or restrictions about dynamic length PDUs. But with parameter SecOCAuthDataFreshnessStartPosition it could be impossible to really use dynamic length in SecOC.

  Additionally in SystemTemplate TPS the constraint constr_3139 talks only about some restrictions when dynamic length IPDUs are used. This would imply dynamic length IPDUs should be possible.

  Without dynamic length support it is also not possible to support securing complete IDPUM Containers.
  One possible solution could be (if dynamic length should not be supported by SecOC) to add a configuration option per IPDUM Container to send always maximum length (padding with 0) to have the static length again.

If dynamic length shall be supported by SecOC a further problem will be the CAN-FD padding.

Agreed solution:

==========
AUTOSAR 4.3.1
==========

_____-
SRS SecOC
_____-

* Add new section next to 6.1.3.5 and new requirement [SRS_SecOC_xxxx1] Support of padding at lower layer modules and dynamic length Authentic I-PDUs.
* Description: The SecOC module shall be applicable for the use cases with padding at lower layer modules and with dynamic length Authentic I-PDUs.
* Rationale: At receiver side, received Secured I-PDU containing dynamic length Authentic I-PDU may also conatin padding bytes (added by lower layer modules of sender side, to fit to bus-specific L-PDU length constraints, e.g. CAN FD and FlexRay). In such case, receivers cannot identify number of bytes / byte position of the received payload.
* Use Case: dynamic length PDU on CAN FD and FlexRay
* Dependencies: [SRS_SecOC_00012]
* Supporting Material: -

Note: According to CM, RS Main should be referred. But upstream requirements in current (R4.3.0) SRS SecOC are from RS Features, and appropriate requirements are not available in RS Main.
(If we use RS Features, at least [RS_BRF_01568] [RS_BRF_01649] [RS_BRF_01712] [RS_BRF_01716] [RS_BRF_01752] [RS_BRF_02035] [RS_BRF_02036] [RS_BRF_02037] could be mapped to this requirements)

_____-
SWS SecOC
_____-

a1) Adapt Figure 4 in sec. 7.1.1.1

Change from
< (Figure of "Secured I-PDU = Authentic I-PDU | Freshness Value | Authenticator")
< Figure 4: Secured I-PDU contents

to
> (Figure of "Secured I-PDU = Secured I-PDU Header (optional) | Authentic I-PDU | Freshness Value (optional) | Authenticator")
> Figure 4: Secured I-PDU contents
==> to be done as RfC # 77807, not handled in this RfC.

a2) Add new requirements regarding the Secured I-PDU Header and related behavior

Add new requirement [SWS_SecOC_xxxx2] to define the Secured I-PDU Header
> The Secured I-PDU Header shall indicate the length of the Authentic I-PDU in bytes. The length of the Header shall be configurable by the parameter SecOCAuthPduHeaderLength.
> ()
> Note: the SecOC supports combined usage of authentication data in a separate message (secured PDU collection) and Secured I-PDU Header. Also the SecOC covers dynamic length Authentic I-PDU.

Add new requirement [SWS_SecOC_xxxx3] for behavior at transmission (construction) of Secured I-PDUs
> For a Tx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall add the Secured I-PDU Header to the Secured I-PDU with the length of the Authentic I-PDU within the Secured I-PDU, to handle dynamic Authentic I-PDU.
> ()
> Note: Primary purpose of this Header is to indicate the position of Freshness Value and Authenticator in Secured I-PDUs with dynamic length Authentic I-PDU.
> Also some buses which cannot select arbitrary length of L-PDU (e.g. CAN FD and FlexRay) require this Header, because the position of Freshness Value and Authenticator is not always at the end of the Secured I-PDU, as lower layer modules (e.g. CanIf and FrIf) may add bus-specific padding bytes after processing at SecOC (then the L-PDU containing the Secured I-PDU with padding will be: Secured I-PDU = Secured I-PDU Header | Authentic I-PDU | Freshness Value | Authenticator | Bus-specific padding).

Add new requirement [SWS_SecOC_xxxx4] for behavior at reception of Secured I-PDUs
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall assume Secured I-PDU Header shall be available in the Secured I-PDU, to handle dynamic Authentic I-PDU.

> ()

Add new requirement [SWS_SecOC_xxxx5] for behavior at reception of Secured I-PDUs, the Header tells it's longer than the maximum length of the PDU
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0 and the length of Authentic I-PDU in the Header is longer than configured length (in case of dynamic length IPdus (containing a dynamical length signal), this value indicates the maximum data length) of the Authentic I-PDU, the SecOC module shall discard the I-PDU. In such case with SecOC_StartOfReception, BUFREQ_E_NOT_OK shall be returned (see [SWS_COMTYPE_00012]).
> ()
> Note: SecOC_RxIndication has no return value.

Add new requirement [SWS_SecOC_xxxx6] for behavior at reception of Secured I-PDUs, the Header tells it's shorter than received I-PDU length (ignoring the padding at the end)
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall process Secured I-PDU Header, Authentic I-PDU (with the length specified by the Header), Freshness Value and Authenticator of the Rx Secured I-PDU. The rest of bytes in the Secured I-PDU shall be discarded.
> ()

a3) Add a configuration parameter to SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPdu, SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection/SecOCRxAuthenticPdu, SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPdu and SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPduCollection/SecOCTxAuthenticPdu to enable/disable Secured I-PDU Header per I-PDU
* SWS Item: ECUC_SecOC_xxxx1
* Name: SecOCAuthPduHeaderLength
* Description:
* This parameter indicates the length (in bytes) of the Secured I-PDU Header in the Secured I-PDU. The length of zero means there's no header in the PDU.
* Multiplicity: 0..1
* Range: 0..4
* Default: 0

a4) (removed)

a8) Update layout definition (construction) for Secured I-PDUs

Change [SWS_SecOC_00037]
from
< [SWS_SecOC_00037]
< The SecOC module shall construct the Secured I-PDU by adding the Freshness Value and the Authenticator to the Authentic I-PDU.
< (SRS_SecOC_00006)
< Note: The Freshness Counter and the Authenticator included as part of the Secured I-PDU may be truncated per configuration specific to the identifier of the Secured I-PDU. The scheme for the Secured I-PDU looks as follows:
< SecuredPDU = AuthenticIPDU | FreshnessValue [SecOCFreshnessValueTxLength] | Authenticator [SecOCAuthInfoTxLength]
to
> [SWS_SecOC_00037]
> The SecOC module shall construct the Secured I-PDU by adding the Secured I-PDU Header (optional), the Freshness Value (optional) and the Authenticator to the Authentic I-PDU.
> The scheme for the Secured I-PDU (includes the order in which the contents are structured in the Secured I-PDU) shall be compliant with below:
> SecuredPDU = SecuredIPDUHeader (optional) | AuthenticIPDU | FreshnessValue [SecOCFreshnessValueTxLength] (optional) | Authenticator [SecOCAuthInfoTxLength]
> (SRS_SecOC_00006)
> Note: The Freshness Counter and the Authenticator included as part of the Secured I-PDU may be truncated per configuration specific to the identifier of the Secured I-PDU. Also Freshness Value may be a part of Authentic I-PDU (see [SWS_SecOC_00219]).
==> to be done as RfC # 77807, not handled in this RfC.

a9) Add new constraints and notes after [SWS_SecOC_00219]:
> [constr_xxxx1] All signals before SecOCAuthDataFreshnessStartPosition within the Secured I-PDU shall have static length.
> Note: SecOC can use a part of the Authentic I-PDU as freshness when SecOCUseAuthDataFreshness=true, only if the part of the Authentic I-PDU to be used as the freshness is always available at same position in the Authentic I-PDU.

> [constr_xxxx2] Any container I-PDU which contains multiple contained I-PDUs shall be set SecOCUseAuthDataFreshness=false.

> Note: For container PDUs, normally it cannot be ensured which PDU will be put in which position (depends on various timing and trigger conditions). Therefore, container I-PDUs with multiple contained I-PDUs cannot have FV within the Authentic I-PDU.

a10) Adapt Figure 5
==> to be done as RfC # 77807, not handled in this RfC.

a11) Adapt SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection/SecOCRxAuthenticPdu and SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPduCollection/SecOCTxAuthenticPdu

Change the description of SecOCRxAuthenticPdu (ECUC_SecOC_00061)
from
< This container specifies the Authetic Pdu that is received by the SecOC module from the PduR.
to
> This container specifies the PDU (that is received by the SecOC module from the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.

Change the description of SecOCTxAuthenticPdu ECUC_SecOC_00072
from
< This container specifies the Authetic Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated.
to
> This container specifies the PDU (that is transmitted by the SecOC module to the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.

———————-
TPS System Template (SysT)
———————-

a5) Add new attribute to SecuredIPdu (Table 6.46) which enables SecOCAuthPduHeaderLength>0
* Attribute: useSecuredPduHeader
* Type: SecuredPduHeaderEnum
* Mul.: 0..1
* Kind: attr
* Desc: This attribute defines the size of the header which is inserted into the

SecuredIPdu. If this attribute is set to anything but noHeader, the SecuredIPdu contains the Secured I-PDU Header to indicate the length of the AuthenticIPdu. The AuthenticIPdu contains the original payload, i.e. the secured data.

SecuredPduHeaderEnum
- noHeader
- securedPduHeader08Bit
- securedPduHeader16Bit
- securedPduHeader32Bit

Desc: Defines the header which will be inserted into the SecuredIPdu.

a6) Change the description of IPduPort.rxSecurityVerification in Table 6.3: IPduPort:
This attribute defines the bypassing of signature authentication or MAC verification in the receiving ECU.
If not defined or set to true the signature authentication or MAC verification shall be performed for the SecuredIPdu.
If set to false the signature authentication or MAC verification shall not be performed for the SecuredIPdu.

Removed [constr_3139].

TPS_SysT_xxxx2: Setting of useSecuredPduHeader attribute
The useSecuredPduHeader shall be set to a value other than noHeader if the length of the payload Pdu is dynamic and is transmitted over a network which may insert padding bytes depending on the length (e.g. CANFD, Flexray).

Add a note below TPS_SysT_xxxx2:

Please note that the dynamic-length Pdu can be an ISignalIPdu that contains a SystemSignal with dynamicLength set to true. In general it is not possible to run diagnostics on fixed-length Pdus. Therefore, there is a probability that at least a subset of DcmIPdus and UserDefinedIPdus can have dynamic length.

a7) Add upstream mapping between useSecuredPduHeader (SysT) and SecOCAuthPduHeaderLength (EcuC) in C.1.4 SecOc Mapping

b4) Add upstream mapping between rxSecurityVerification (SysT) and SecOCSecuredRxPduVerification (EcuC) in C.1.4 SecOc Mapping
Mapping rule: SecOCSecuredRxPduVerification is True if rxSecurityVerification is not defined, otherwise SecOCSecuredRxPduVerification = rxSecurityVerification

———————-

SRS SecOC
————————-

* Add new section next to 6.1.3.5 (or 6.2.1.1) and new requirement [SRS_SecOC_xxxx2] Support of capability to extract Authentic I-PDU without Authentication
* Description: The SecOC module shall be capable to extract Authentic I-PDU from Secured I-PDU, without Authentication.
* Rationale: SecOC can be used as an extractor of Authentic I-PDU from Secured I-PDU, to enable low latency GW behavior when a part of downstream communication clusters doesn't require authentication of PDUs.
* Use Case: Gateway
* Dependencies: [SRS_SecOC_00025]
* Supporting Material: -

Note: According to CM, RS Main should be referred. But upstream requirments in current (R4.3.0) SRS SecOC are from RS Features, and appropriate requirements are not available in RS Main.
(If we use RS Features, at least [RS_BRF_02035] [RS_BRF_02036] [RS_BRF_02037] could be mapped to this requirements)

————————-
SWS SecOC
————————-

b1) Remove [constr_3139] (not [constr_3193] – sorry, constr_3193 is typo in my comment # 10)

b2) Add new requirements regarding skipped authentication behavior at SecOC (just remove FV/MAC from Secured I-PDU)

* Add new section "Extracting Authentic I-PDU without Authentication at SecOC" or "Skipping Authentication for Secured I-PDUs at SecOC"
* Add new requirement [SWS_SecOC_xxxx7] for behavior of SecOC at reception of Secured I-PDUs without Authentication
> For a Rx Secured I-PDU with SecOCSecuredRxPduVerification=false, the SecOC module shall extract the Authentic I-PDU using the length specified by the Secured I-PDU Header without Authentication.
> ()

b3) Add a configuration parameter to SecOC/SecOCRxPduProcessing/Sec-

OCRxSecuredPduLayer/SecOCRxSecuredPdu and SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection to control authentication behavior at SecOC

* SWS Item: ECUC_SecOC_xxxx3
* Name: SecOCSecuredRxPduVerification
* Description: This parameter defines whether the signature authentication or MAC verification shall be performed on this Secured I-PDU. If set to false, the SecOC module extracts the Authentic I-PDU from the Secured I-PDU without verification.
* Multiplicity: 1
* Type: EcucBooleanParamDef
* Default value: false
* Post-Build Variant Value: true
* Value Configuration Class:
* Pre-compile time: X All Variants
* Scope / Dependency: scope: local
–Last change on issue 77336 comment 69–

BW-C-Level:

Application	Specification	Bus
1	4	4

1.6 Specification Item constr_3364

Trace References:

none

Content:

The header length in bits specified by BufferProperties.headerLength shall be a multiple of 8.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73003: Inconsequent usage of bits and bytes for BufferProperties.headerLength

  Problem description:

  _____

  Name: Per-Ola Geisler
  Phone: +46 31 725 49 14
  Role: Developer

Description/Motivation:
1) If I look in the in the General Specification of Transformers it is stated that the header length shall be specified in bits (see Table A.3: BufferProperties).
If I look in System Template, the examples (see Listing 7.1 and Listing 7.2) indicates that header length shall be specified in bytes.
I could find a similar reported ticket concerning the Specification of Module E2E Transformer (RfC # 69897), but it did not consider the System Template.

2) If it shall be specified in bits, how should the Rte requirement below be implemented, when the headerlength is not divisible by 8 (truncate or increase to nearest byte), since it is only possible to address whole bytes:

[SWS_Rte_08536] The RTE shall consider the headerLength information in the BufferProperties if inPlace in the BufferProperties is set:
- On the sending side (transformation) the RTE shall increase the buffer from the beginning by the size given in headerLength.
- On the receiving side (retransformation) the RTE shall decrease the buffer from the beginning by the size given in headerLength.

Was there already a decision?

Agreed solution:

Introduce a new spec item:3

Self-contained copy-paste-ready Solution Proposal (with requirement IDs / specification items):

System Template:
introduce a new constraint (below [constr_3126]:
The header length in bits specified by headerLength shall be a multiple of 8.

Listing 7.1:
change "<HEADER-LENGTH>8</HEADER-LENGTH>" to "<HEADER-LENGTH>64</HEADER-LENGTH>".
Listing 7.2:

change "<HEADER-LENGTH>5</HEADER-LENGTH>" to "<HEADER-LENGTH>40</HEADER-LENGTH>".

AUTOSAR_SWS_E2ETransformer:
Table 7.19:
change table row "BufferProperties.headerLength | 12 | 12 bits is the length of the E2E profile 1C header." to ""BufferProperties.headerLength | 16 | 16 bits is the length of the E2E profile 1C header."
–Last change on issue 73003 comment 17–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.7 Specification Item constr_3365

Trace References:

none

Content:

A mix of EthernetPhysicalChannels with different Identifiable.category values within an EthernetCluster is currently not supported by AUTOSAR.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76287: TPS_SYST_02160 cannot be implemented

  Problem description:

  TPS_SYST_02160 states a negative requirement:

  EthernetPhysicalChannels with different category values
  are not allowed within an EthernetCluster d A mix of EthernetPhysicalChannels with different category values within an EthernetCluster is currently not supported by AUTOSAR. c()

  This cannot be implemented. It can only be checked by a tool and therefore needs be converted into a constraint.

  Agreed solution:

  Change Spec Items TPS_SYST_02160 to a new constraint:

[constr_xxxxx] EthernetPhysicalChannels with different category values
are not allowed within an EthernetCluster d A mix of EthernetPhysicalChannels
with different category values within an EthernetCluster is currently
not supported by AUTOSAR. c()
–Last change on issue 76287 comment 4–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.8 Specification Item constr_3373

Trace References:

none

Content:

A CommunicationConnector shall only be referenced by at most one PhysicalChannel.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76529: Connection between GlobalTimeDomain and VLANs

  Problem description:

  The connection from GobalTimeDomain to the VLANs (PhysicalChannel:s) are made from the GlobalTimeMaster(GlobalTimeSlave to the PhysicalChannels via CommunicationConnector:s

  It is possible that the user only creates ONE CommunicationConnector for the different VLANs (PhyscialChannles), because they are anyway connected to same CommunicationController. This will mean that the GlobalTimeMaster/-GlobalTimeSlave will reference this CommunicationConnector. When an ECU configuration tool uses the UTM rule to transform to ECUC, it will likely get the wrong VLAN
  See attached example.

  Suggestion:
  Add a constraint that the must be one CommunicationConnector for each VLAN and untagged (PhysicalChannl). Or is there a way to model it such that a constraint can be avoided?

  Agreed solution:

The source multiplicity of the PhysicalChannel->CommunicationConnector shall be replaced by a constraint:

[constr_xxxx] Limitation on the number of PhysicalChannels that are referencing a CommunicationConnector
A CommunicationConnector shall only be referenced by at most one PhysicalChannel.
–Last change on issue 76529 comment 4–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.9 Specification Item constr_3378

Trace References:

none

Content:

In a given instance of AliasNameSet in the bound system there shall be at most one AliasNameSet.aliasName per FlatInstanceDescriptor.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77367: Change a statement in AliasNameSet class table to a formal constraint

  Problem description:

  This is a follow up of # 77224.

  The following statement in class table AliasNameSet shall be changed to a formal constraint in the specification:
  "In a given instance of AliasNameSet in the bound system there shall be at most one aliasName per FlatInstanceDescriptor."

  Agreed solution:

  Remove the statement "In a given instance of AliasNameSet in the bound system there shall be at most one
  aliasName per FlatInstanceDescriptor." from the AliasNameSet class table and introduce a corresponding constraint in the specification.

  BW-C-Level:

Application	Specification	Bus
1	4	1

1.10 Specification Item constr_3379

Trace References:

none

Content:

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/Consumed ServiceInstance may only be defined in the ApplicationEndpoint aggregated by one of these SocketAddress entries.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77037: Clarification on the use of SocketAddress instances with the same attributes in different SocketConnectionBundles

  Problem description:

  The use of SocketAddress instances in the EcuExtracts shall be clarified in the situation where one ECU represents a Server in one SocketConnectionBundle and a Client in another SocketConnectionBundle, but uses the same ipAddress, portAddress and applicationEndpoint for communication in both bundles. In other words, is it required to reference the same instance of the SocketAddress in both cases, first in the role serverPort and second in the role clientPort in the scope of the corresponding SocketConnection? Or is it required to use two identical SocketAddress instances?

  Experts in the field I talked to fully disagree, both on the same premises of invalid ECU configuration that would be derived from the EcuExtract in these cases (SoAdSocketConnectionGroups). Perhaps one source of confusion could be related to the fact that it may not be entirely clear whether the following sentence in the mapping rule description of the SoAdSocketConnectionGroup applies to Server and Client ECUs, or Client ECUs only?

  "Please note that the same SoAdSocketConnectionGroup shall be used for all SocketConnections with the same SoAdSocketLocalAddress, SoAdSocketLocalPort and TP."

Also by reading this sentence, it seems to me that it is the tuple of ipAddress, portAddress and tpConfiguration that is important for the derivation of the SoAdSocketConnectionGroups in the ECUC, not the exact instance of the SocketAddress in the EcuExtract.

Agreed solution:

The following constraint shall be added in the System Template below the ProvidedServiceInstance and ConsumedServiceInstance tables:

constr_xxx1: Multiple SocketAddress entries with the same IP Address, Protocol and Port in the context of a given Ecu Instance [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.]

Rationale for constr_xxx1: There can be only one representation of the ProvidedServiceInstance/ConsumedServiceInstance using the given IP Address, Protocol and Port# in the Sd module configuration in the context of a given Ecu Instance. Therefore, defining ProvidedServiceInstance/ConsumedServiceInstance in more than one ApplicationEndPoint would in this case require a merge of potentially different attribute values of the ProvidedServiceInstances and/or ConsumedServiceInstances in the System Template, the situation avoided by this constraint.
–Last change on issue 77037 comment 5–

BW-C-Level:

Application	Specification	Bus
1	3	1

1.11 Specification Item constr_3383

Trace References:

none

Content:

The following values of the attribute Identifiable.category of meta-class GeneralPurpose Connection are reserved by the AUTOSAR standard:

- XcpChannel

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74995: [XCP] Clean up Chapters 8 and 10

  Problem description:

  XCP lacks the possibility to associate Rx-PDUs with Tx-PDUs of the same bus. Besides, a reference to a ComMChannel is missing in the EcuC that is used as an argument to Xcp_SetTransmissionMode() (which by the way is in the wrong section).

  Agreed solution:

  SWS XCP:
  Regarding the XCP configuration must be adapted such that there is a link between the Rx and Tx PDUs (XcpRxPdu/XcpTxPdu) that are used for request/response from one master.
  - Xcp_SetTransmissionMode() has a ComM channel as an argument, but the XCP does not know which PDUs belong to which ComM channel (or at least that is quite complicated to find out).

  The PS
  - Introduction of a new container (XcpCommunicationChannel) which references the corresponding Tx and Rx PDU (Tx PDU is mandatory, Rx PDU is optional) and the ComM channel they belong to.
  - Move Xcp_SetTransmissionMode() Spec Items SWS_Xcp_00844, SWS_Xcp_00848, SWS_Xcp_00849 and SWS_Xcp_00850 from section 8.4 to section 8.3.

  Create new CR for ECUC XML:
  Create new XCP configuration container:
  Container name: XcpCommunicationChannel
  Container Parameter:
  - XcpChannelTxPduRef: TX PDU (XcpTxPdu) reference (mandatory)
  - XcpChannelRxPduRef: RX PDU (XcpRxPdu) reference (optional)
  - XcpComMChannel: ComM channel (ComMChannel) reference (mandatory): which define what ComM channel which XCP belong to.

  SysT:

  - Introduce a new ARElement "GeneralPurposeConnection" that references to * PduTriggerings.
  - Introduce a new category "XcpChannel" for "GeneralPurposeConnection" and define that in case of an "XcpChannel" the "GeneralPurposeConnection" is allowed

to reference exactly two PduTriggerings (one TX and one RX Pdu).
- Introduce a constraint that defines that the PduTriggerings that are referenced by a "GeneralPurposeConnection" are defined on the same PhysicalChannel.
- Introduce a constraint that defines that the PduTriggerings that are referenced by "GeneralPurposeConnection" of category "XcpChannel" shall refer to Pdus of type "GeneralPurposeIPdu" with category "XCP"

In addition the following changes shall be done in the specification:
- Remove footnote on the first page of Section 6.3.
- Rework the first paragraph of Section 6.3 to the following: "The PDU Router is responsible only for the routing of IPdus (i.e., other types of Pdus are not routed by the PDU Router)."
- Remove footnote in Section 6.8.2.
–Last change on issue 74995 comment 25–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.12 Specification Item constr_3384

Trace References:

none

Content:

The PduTriggerings that are referenced by the GeneralPurposeConnection in the role GeneralPurposeConnection.pduTriggering shall be defined on the same PhysicalChannel.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74995: [XCP] Clean up Chapters 8 and 10

  Problem description:

  XCP lacks the possibility to associate Rx-PDUs with Tx-PDUs of the same bus. Besides, a reference to a ComMChannel is missing in the EcuC that is used as an argument to Xcp_SetTransmissionMode() (which by the way is in the wrong section).

  Agreed solution:

  SWS XCP:
  Regarding the XCP configuration must be adapted such that there is a link between

the Rx and Tx PDUs (XcpRxPdu/XcpTxPdu) that are used for request/response from one master.
- Xcp_SetTransmissionMode() has a ComM channel as an argument, but the XCP does not know which PDUs belong to which ComM channel (or at least that is quite complicated to find out).

The PS
- Introduction of a new container (XcpCommunicationChannel) which references the corresponding Tx and Rx PDU (Tx PDU is mandatory, Rx PDU is optional) and the ComM channel they belong to.
- Move Xcp_SetTransmissionMode() Spec Items SWS_Xcp_00844, SWS_Xcp_00848, SWS_Xcp_00849 and SWS_Xcp_00850 from section 8.4 to section 8.3.

Create new CR for ECUC XML:
Create new XCP configuration container:
Container name: XcpCommunicationChannel
Container Parameter:
- XcpChannelTxPduRef: TX PDU (XcpTxPdu) reference (mandatory)
- XcpChannelRxPduRef: RX PDU (XcpRxPdu) reference (optional)
- XcpComMChannel: ComM channel (ComMChannel) reference (mandatory): which define what ComM channel which XCP belong to.

SysT:

- Introduce a new ARElement "GeneralPurposeConnection" that references to * PduTriggerings.
- Introduce a new category "XcpChannel" for "GeneralPurposeConnection" and define that in case of an "XcpChannel" the "GeneralPurposeConnection" is allowed to reference exactly two PduTriggerings (one TX and one RX Pdu).
- Introduce a constraint that defines that the PduTriggerings that are referenced by a "GeneralPurposeConnection" are defined on the same PhysicalChannel.
- Introduce a constraint that defines that the PduTriggerings that are referenced by "GeneralPurposeConnection" of category "XcpChannel" shall refer to Pdus of type "GeneralPurposeIPdu" with category "XCP"

In addition the following changes shall be done in the specification:
- Remove footnote on the first page of Section 6.3.
- Rework the first paragraph of Section 6.3 to the following: "The PDU Router is responsible only for the routing of IPdus (i.e., other types of Pdus are not routed by the PDU Router)."
- Remove footnote in Section 6.8.2.
–Last change on issue 74995 comment 25–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.13 Specification Item constr_3385

Trace References:

none

Content:

In case that the Identifiable.category of meta-class GeneralPurposeConnection is set to the value XcpChannel the GeneralPurposeConnection is allowed to reference exactly two PduTriggerings in the role GeneralPurposeConnection.pduTriggering.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74995: [XCP] Clean up Chapters 8 and 10

  Problem description:

  XCP lacks the possibility to associate Rx-PDUs with Tx-PDUs of the same bus. Besides, a reference to a ComMChannel is missing in the EcuC that is used as an argument to Xcp_SetTransmissionMode() (which by the way is in the wrong section).

  Agreed solution:

  SWS XCP:
  Regarding the XCP configuration must be adapted such that there is a link between the Rx and Tx PDUs (XcpRxPdu/XcpTxPdu) that are used for request/response from one master.
  - Xcp_SetTransmissionMode() has a ComM channel as an argument, but the XCP does not know which PDUs belong to which ComM channel (or at least that is quite complicated to find out).

  The PS
  - Introduction of a new container (XcpCommunicationChannel) which references the corresponding Tx and Rx PDU (Tx PDU is mandatory, Rx PDU is optional) and the ComM channel they belong to.
  - Move Xcp_SetTransmissionMode() Spec Items SWS_Xcp_00844, SWS_Xcp_00848, SWS_Xcp_00849 and SWS_Xcp_00850 from section 8.4 to section 8.3.

Create new CR for ECUC XML:
Create new XCP configuration container:
Container name: XcpCommunicationChannel
Container Parameter:
- XcpChannelTxPduRef: TX PDU (XcpTxPdu) reference (mandatory)
- XcpChannelRxPduRef: RX PDU (XcpRxPdu) reference (optional)
- XcpComMChannel: ComM channel (ComMChannel) reference (mandatory): which define what ComM channel which XCP belong to.

SysT:

- Introduce a new ARElement "GeneralPurposeConnection" that references to * PduTriggerings.
- Introduce a new category "XcpChannel" for "GeneralPurposeConnection" and define that in case of an "XcpChannel" the "GeneralPurposeConnection" is allowed to reference exactly two PduTriggerings (one TX and one RX Pdu).
- Introduce a constraint that defines that the PduTriggerings that are referenced by a "GeneralPurposeConnection" are defined on the same PhysicalChannel.
- Introduce a constraint that defines that the PduTriggerings that are referenced by "GeneralPurposeConnection" of category "XcpChannel" shall refer to Pdus of type "GeneralPurposeIPdu" with category "XCP"

In addition the following changes shall be done in the specification:
- Remove footnote on the first page of Section 6.3.
- Rework the first paragraph of Section 6.3 to the following: "The PDU Router is responsible only for the routing of IPdus (i.e., other types of Pdus are not routed by the PDU Router)."
- Remove footnote in Section 6.8.2.
–Last change on issue 74995 comment 25–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.14 Specification Item constr_3386

Trace References:

none

Content:

In case that the Identifiable.category of meta-class GeneralPurposeConnection is set to the value XcpChannel the GeneralPurposeConnection is allowed to reference PduTriggerings of GeneralPurposeIPdus with category XCP.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74995: [XCP] Clean up Chapters 8 and 10

  Problem description:

  XCP lacks the possibility to associate Rx-PDUs with Tx-PDUs of the same bus. Besides, a reference to a ComMChannel is missing in the EcuC that is used as an argument to Xcp_SetTransmissionMode() (which by the way is in the wrong section).

  Agreed solution:

  SWS XCP:
  Regarding the XCP configuration must be adapted such that there is a link between the Rx and Tx PDUs (XcpRxPdu/XcpTxPdu) that are used for request/response from one master.
  - Xcp_SetTransmissionMode() has a ComM channel as an argument, but the XCP does not know which PDUs belong to which ComM channel (or at least that is quite complicated to find out).

  The PS
  - Introduction of a new container (XcpCommunicationChannel) which references the corresponding Tx and Rx PDU (Tx PDU is mandatory, Rx PDU is optional) and the ComM channel they belong to.
  - Move Xcp_SetTransmissionMode() Spec Items SWS_Xcp_00844, SWS_Xcp_00848, SWS_Xcp_00849 and SWS_Xcp_00850 from section 8.4 to section 8.3.

  Create new CR for ECUC XML:
  Create new XCP configuration container:
  Container name: XcpCommunicationChannel
  Container Parameter:
  - XcpChannelTxPduRef: TX PDU (XcpTxPdu) reference (mandatory)
  - XcpChannelRxPduRef: RX PDU (XcpRxPdu) reference (optional)
  - XcpComMChannel: ComM channel (ComMChannel) reference (mandatory): which define what ComM channel which XCP belong to.

  SysT:

  - Introduce a new ARElement "GeneralPurposeConnection" that references to * PduTriggerings.

- Introduce a new category "XcpChannel" for "GeneralPurposeConnection" and define that in case of an "XcpChannel" the "GeneralPurposeConnection" is allowed to reference exactly two PduTriggerings (one TX and one RX Pdu).
- Introduce a constraint that defines that the PduTriggerings that are referenced by a "GeneralPurposeConnection" are defined on the same PhysicalChannel.
- Introduce a constraint that defines that the PduTriggerings that are referenced by "GeneralPurposeConnection" of category "XcpChannel" shall refer to Pdus of type "GeneralPurposeIPdu" with category "XCP"

In addition the following changes shall be done in the specification:
- Remove footnote on the first page of Section 6.3.
- Rework the first paragraph of Section 6.3 to the following: "The PDU Router is responsible only for the routing of IPdus (i.e., other types of Pdus are not routed by the PDU Router)."
- Remove footnote in Section 6.8.2.
–Last change on issue 74995 comment 25–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.15 Specification Item constr_3399

Trace References:

none

Content:

If the SecureCommunicationProps.securedAreaOffset is defined then the SecureCommunicationProps.securedAreaLength shall be defined as well and vice versa.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77090: [SecOC] Configuration of secured area within a Pdu

  Problem description:

  Currently it is not possible to configure a range/area within one Pdu which shall be secured. This possibility shall be added

  Agreed solution:

  SWS_SecOC:
  See attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4500

~change range from 1..2^32-1 to 0..2^32-1

TPS_SystemTemplate:
add the following optional attributes to SecureCommunicationProps:
- securedAreaOffset (PositiveInteger) - This attribute defines the start position (offset in byte) of the area within the payload Pdu which will be secured
- securedAreaLength (PositiveInteger) - This attribute defines the length in bytes of the area within the payload Pdu which will be secured

Add the following specification item into chapter 6.3.2 SecuredIPdu

[TPS_SYST_0XXX1] Secured Area in payload Pdu
The area within the payload Pdu that is secured is specified by the securedAreaOffset and securedAreaLength. In case that these two attributes are not configured the complete
payload Pdu is secured.

[constr_xxx1] Existence of securedAreaOffset and securedAreaLength
If the securedAreaOffset is defined then the securedAreaLength shall be defined as well and vice versa.
–Last change on issue 77090 comment 29–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.16 Specification Item constr_3400

Trace References:

none

Content:

Usage of SdClientConfig attributes in ConsumedServiceInstance and ConsumedEventGroup shall follow the restrictions given in [REF table_3a_SdClientAttributes].

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

Name: Robert Sakretz
Phone:
Role: WP-M

Description/Motivation:

The classes SdServerConfig and SdClientConfig are aggregated two times:
ProvidedServiceInstance.sdServerConfig
EventHandler.sdServerConfig
ConsumedServiceInstance.sdClientConfig
ConsumedEventGroup.sdClientConfig

This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
EventHandler.sdServerConfig.serverServiceMajorVersion

In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the

initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.

- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.17 Specification Item constr_3401

Trace References:

none

Content:

Usage of SdServerConfig attributes in ProvidedServiceInstance and EventHandler shall follow the restrictions given in [REF table_3a_SdServerAttributes].

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  ———————————
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  ———————————
  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
  ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
  EventHandler.sdServerConfig.serverServiceMajorVersion

  In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

  My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
  So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

  Agreed solution:

  Add the following tables into the specification as explanation (see also attachment):
  https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

  Add the following explanations as specification items in addition to the new

tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values

expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.18 Specification Item constr_3402

Trace References:

none

Content:

For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu.contained PduTriggering with ContainerIPdu.ContainerIPdu.headerType = ContainerIPduHeaderTypeEnum.noHeader the IPdu.IPdu.containedIPduProps.ContainedIPduProps.offset shall be defined.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76543: [IpduM] Container Pdu not usable for CAN-FD

  Problem description:

  With the CAN-FD extension, the Container PDU concept has been introduced. Unfortunately this it not usable in a meaningful way for CAN-FD due to the 4-byte header for each contained PDU of a Container Pdu.

  Agreed solution:

  Proposed solution for the SWS IPduM:

  Extend the enumeration of IpduMContainerHeaderSize by:
  - IPDUM_HEADERTYPE_LONG
  - IPDUM_HEADERTYPE_SHORT
  - IPDUM_HEADERTYPE_NONE

  The detailed proposed solution is contained here: https://svn.autosar.org/repos/work/09_WorkPackages/WP-A2/70_InternalDocuments/RfCs/76543-AUTOSAR_SWS_IPDUMultiplexer.doc

  ==========
  Proposed solution TPS_SystemTemplate:

Extend the enumeration of ContainerIPduHeaderTypeEnum by:
- noHeader: No Header is used and the the location of each containedPdu in the ContainerPdu is statically configured.

Add the following optional attributes to ContainedIPduProps
- offset (PositiveInteger): Byte offset that describes the location of the ContainedPdu in the ContainerPdu if no header is used.
- updateIndicationBitPosition (PositiveInteger): The updateIndicationBit specifies the bit location of ContainedIPdu Update-Bit in the Container PDU. It indicates to the receivers that the ContainedIPdu in the ContainerIPdu was updated.

Add the following optional attributes to ContainerIPdu
- unusedBitPattern (PositiveInteger): IPduM fills not updated areas of the ContainerPdu with this byte-pattern.

Rewrite the Introduction of chapter 6.3.1 ContainerIPdu

For each IPdu which is put inside a ContainerIPdu, a header may be provided which determines which IPdu is contained (ContainedIPduProps.headerIdLongHeader or headerIdShortHeader and what the size of that IPdu is (DLC during runtime). With this header mode the receivers are able to extract the individual contained IPdus again. As an alternative option to the usage of headers a statically configured layout of IPdus in the ContainerIPdu is supported.

Introduce the following constraints:

[constr_xxxx1] Mandatory offset if noHeader is used
For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu.containedPduTriggering with ContainerIPdu.headerType = noHeader the IPdu.containedIPduProps.offset shall be defined.

[constr_xxxx2] 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_xxxx3] 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_xxxx4] Only the last contained IPdu (according to the ContainedIPduProps.offset) of a ContainerIPdu with static container layout (i.e., a ContainerIPdu with ContainerIPduHeaderTypeEnum set to noHeader) may be a dynamic length PDU (i.e, a PDU that at runtime may exhibit a length different from the one statically configured via Pdu.length of the respective Pdu). All other contained PDUs of a ContainerIPdu with static container layout have to be static length PDUs.

Adapt the following spec items to:

[TPS_SYST_02098] Header id and header type of a contained IPdu
A contained IPdu shall always have the same headerId per header type (long or short header), regardless in which ContainerIPdu it is collected. If noHeader is set then the contained Ipdu does not need to have a headerId.

[TPS_SYST_02100] Relation between ContainerIPdu and contained IPdus on receiver side
On receiver side, it is not necessarily required to statically define which IPdus may be contained inside a ContainerIPdu if the header mode is used. Thus it would be possible to update the senders of ContainerIPdus and put different or additional IPdus inside.

Rework descriptions of the following classes and attributes in the model:

ContainerIPdu class description: Allows to collect several IPdus in one ContainerIPdu based on the headerType
ContainerIPdu.headerType attribute description: Defines whether and which header type is used (header id and length).

==========
Proposed solution SRS_IpduM:

- rename Chapter 6.1.4 to Dynamic I-PDU to Container Mapping
rename [SRS_IpduM_02820] to Dynamic I-PDU Mapping
add a dependency: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_0XXXX) but no mixture.
- add new Chapter Static I-PDU to Container Mapping
- add new SRS requirement: Static I-PDU Mapping
Type: Draft
Description: Multiple I-PDUs shall be mappable to a static position into a Container-PDU. If an I-PDU is mapped statically to a Container it is always transported in the

same position of the Container-PDU.
Rationale: Support scenarios of Container Mapping where no header shall be used to reduce bandwidth in case always all contained PDU shall be sent.
Use Case: Efficient transmission of small PDUs on high bandwidth busses
Dependencies: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_02820) but no mixture.
–Last change on issue 76543 comment 100–

BW-C-Level:

Application	Specification	Bus
1	2	1

1.19 Specification Item constr_3403

Trace References:

none

Content:

If the ContainerIPdu.ContainerIPdu.headerType is set to ContainerIPduHeaderTypeEnum.noHeader then the ContainerIPdu.ContainerIPdu.rxAcceptContainedIPdu attribute value shall be set to RxAcceptContainedIPduEnum.acceptConfigured.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76543: [IpduM] Container Pdu not usable for CAN-FD

  Problem description:

  With the CAN-FD extension, the Container PDU concept has been introduced. Unfortunately this it not usable in a meaningful way for CAN-FD due to the 4-byte header for each contained PDU of a Container Pdu.

  Agreed solution:

  Proposed solution for the SWS IPduM:

  Extend the enumeration of IpduMContainerHeaderSize by:
  - IPDUM_HEADERTYPE_LONG
  - IPDUM_HEADERTYPE_SHORT
  - IPDUM_HEADERTYPE_NONE

  The detailed proposed solution is contained here: https://svn.autosar.org/repos/work/09_WorkPackages/WP-

A2/70_InternalDocuments/RfCs/76543-AUTOSAR_SWS_IPDUMultiplexer.doc

==========
Proposed solution TPS_SystemTemplate:

Extend the enumeration of ContainerIPduHeaderTypeEnum by:
- noHeader: No Header is used and the the location of each containedPdu in the ContainerPdu is statically configured.

Add the following optional attributes to ContainedIPduProps
- offset (PositiveInteger): Byte offset that describes the location of the ContainedPdu in the ContainerPdu if no header is used.
- updateIndicationBitPosition (PositiveInteger): The updateIndicationBit specifies the bit location of ContainedIPdu Update-Bit in the Container PDU. It indicates to the receivers that the ContainedIPdu in the ContainerIPdu was updated.

Add the following optional attributes to ContainerIPdu
- unusedBitPattern (PositiveInteger): IPduM fills not updated areas of the ContainerPdu with this byte-pattern.

Rewrite the Introduction of chapter 6.3.1 ContainerIPdu

For each IPdu which is put inside a ContainerIPdu, a header may be provided which determines which IPdu is contained (ContainedIPduProps.headerIdLongHeader or headerIdShortHeader and what the size of that IPdu is (DLC during runtime). With this header mode the receivers are able to extract the individual contained IPdus again. As an alternative option to the usage of headers a statically configured layout of IPdus in the ContainerIPdu is supported.

Introduce the following constraints:

[constr_xxxx1] Mandatory offset if noHeader is used
For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu.containedPduTriggering with ContainerIPdu.headerType = noHeader the IPdu.containedIPduProps.offset shall be defined.

[constr_xxxx2] 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_xxxx3] 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_xxxx4] Only the last contained IPdu (according to the ContainedIPduProps.offset) of a ContainerIPdu with static container layout (i.e., a ContainerIPdu with ContainerIPduHeaderTypeEnum set to noHeader) may be a dynamic length PDU (i.e, a PDU that at runtime may exhibit a length different from the one statically configured via Pdu.length of the respective Pdu). All other contained PDUs of a ContainerIPdu with static container layout have to be static length PDUs.

Adapt the following spec items to:

[TPS_SYST_02098] Header id and header type of a contained IPdu
A contained IPdu shall always have the same headerId per header type (long or short header), regardless in which ContainerIPdu it is collected. If noHeader is set then the contained Ipdu does not need to have a headerId.

[TPS_SYST_02100] Relation between ContainerIPdu and contained IPdus on receiver side
On receiver side, it is not necessarily required to statically define which IPdus may be contained inside a ContainerIPdu if the header mode is used. Thus it would be possible to update the senders of ContainerIPdus and put different or additional IPdus inside.

Rework descriptions of the following classes and attributes in the model:

ContainerIPdu class description: Allows to collect several IPdus in one ContainerIPdu based on the headerType
ContainerIPdu.headerType attribute description: Defines whether and which header type is used (header id and length).

==========
Proposed solution SRS_IpduM:

- rename Chapter 6.1.4 to Dynamic I-PDU to Container Mapping
rename [SRS_IpduM_02820] to Dynamic I-PDU Mapping
add a dependency: A Container-PDU shall either support static or dynamic mapping

(see SRS_IpduM_0XXXX) but no mixture.
- add new Chapter Static I-PDU to Container Mapping
- add new SRS requirement: Static I-PDU Mapping
Type: Draft
Description: Multiple I-PDUs shall be mappable to a static position into a Container-PDU. If an I-PDU is mapped statically to a Container it is always transported in the same position of the Container-PDU.
Rationale: Support scenarios of Container Mapping where no header shall be used to reduce bandwidth in case always all contained PDU shall be sent.
Use Case: Efficient transmission of small PDUs on high bandwidth busses
Dependencies: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_02820) but no mixture.
–Last change on issue 76543 comment 100–

BW-C-Level:

Application	Specification	Bus
1	2	1

1.20 Specification Item constr_3404

Trace References:

none

Content:

ContainedIPduProps.ContainedIPduProps.updateIndicationBitPosition is only allowed to be set to a value if the ContainerIPdu.headerType of the ContainerIPdu that contains the IPdu with IPdu.containedIPduProps is set to ContainerIPduHeaderTypeEnum.noHeader.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76543: [IpduM] Container Pdu not usable for CAN-FD

  Problem description:

  With the CAN-FD extension, the Container PDU concept has been introduced. Unfortunately this it not usable in a meaningful way for CAN-FD due to the 4-byte header for each contained PDU of a Container Pdu.

  Agreed solution:

  Proposed solution for the SWS IPduM:

  Extend the enumeration of IpduMContainerHeaderSize by:

- IPDUM_HEADERTYPE_LONG
- IPDUM_HEADERTYPE_SHORT
- IPDUM_HEADERTYPE_NONE

The detailed proposed solution is contained here: https://svn.autosar.org/repos/work/09_WorkPackages/WP-A2/70_InternalDocuments/RfCs/76543-AUTOSAR_SWS_IPDUMultiplexer.doc

==========

Proposed solution TPS_SystemTemplate:

Extend the enumeration of ContainerIPduHeaderTypeEnum by:
- noHeader: No Header is used and the the location of each containedPdu in the ContainerPdu is statically configured.

Add the following optional attributes to ContainedIPduProps
- offset (PositiveInteger): Byte offset that describes the location of the ContainedPdu in the ContainerPdu if no header is used.
- updateIndicationBitPosition (PositiveInteger): The updateIndicationBit specifies the bit location of ContainedIPdu Update-Bit in the Container PDU. It indicates to the receivers that the ContainedIPdu in the ContainerIPdu was updated.

Add the following optional attributes to ContainerIPdu
- unusedBitPattern (PositiveInteger): IPduM fills not updated areas of the ContainerPdu with this byte-pattern.

Rewrite the Introduction of chapter 6.3.1 ContainerIPdu

For each IPdu which is put inside a ContainerIPdu, a header may be provided which determines which IPdu is contained (ContainedIPduProps.headerIdLongHeader or headerIdShortHeader and what the size of that IPdu is (DLC during runtime). With this header mode the receivers are able to extract the individual contained IPdus again. As an alternative option to the usage of headers a statically configured layout of IPdus in the ContainerIPdu is supported.

Introduce the following constraints:

[constr_xxxx1] Mandatory offset if noHeader is used
For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu.containedPduTriggering with ContainerIPdu.headerType = noHeader the

IPdu.containedIPduProps.offset shall be defined.

[constr_xxxx2] 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_xxxx3] 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_xxxx4] Only the last contained IPdu (according to the ContainedIPduProps.offset) of a ContainerIPdu with static container layout (i.e., a ContainerIPdu with ContainerIPduHeaderTypeEnum set to noHeader) may be a dynamic length PDU (i.e, a PDU that at runtime may exhibit a length different from the one statically configured via Pdu.length of the respective Pdu). All other contained PDUs of a ContainerIPdu with static container layout have to be static length PDUs.

Adapt the following spec items to:

[TPS_SYST_02098] Header id and header type of a contained IPdu
A contained IPdu shall always have the same headerId per header type (long or short header), regardless in which ContainerIPdu it is collected. If noHeader is set then the contained Ipdu does not need to have a headerId.

[TPS_SYST_02100] Relation between ContainerIPdu and contained IPdus on receiver side
On receiver side, it is not necessarily required to statically define which IPdus may be contained inside a ContainerIPdu if the header mode is used. Thus it would be possible to update the senders of ContainerIPdus and put different or additional IPdus inside.

Rework descriptions of the following classes and attributes in the model:

ContainerIPdu class description: Allows to collect several IPdus in one ContainerIPdu based on the headerType
ContainerIPdu.headerType attribute description: Defines whether and which header type is used (header id and length).

==========
Proposed solution SRS_IpduM:

- rename Chapter 6.1.4 to Dynamic I-PDU to Container Mapping
rename [SRS_IpduM_02820] to Dynamic I-PDU Mapping
add a dependency: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_0XXXX) but no mixture.
- add new Chapter Static I-PDU to Container Mapping
- add new SRS requirement: Static I-PDU Mapping
Type: Draft
Description: Multiple I-PDUs shall be mappable to a static position into a Container-PDU. If an I-PDU is mapped statically to a Container it is always transported in the same position of the Container-PDU.
Rationale: Support scenarios of Container Mapping where no header shall be used to reduce bandwidth in case always all contained PDU shall be sent.
Use Case: Efficient transmission of small PDUs on high bandwidth busses
Dependencies: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_02820) but no mixture.
–Last change on issue 76543 comment 100–

BW-C-Level:

Application	Specification	Bus
1	2	1

1.21 Specification Item constr_3405

Trace References:

none

Content:

Only the last contained IPdu (according to the ContainedIPduProps.ContainedIPdu Props.offset) of a ContainerIPdu with static container layout (i.e., a ContainerIPdu with ContainerIPdu.headerType set to ContainerIPduHeaderTypeEnum.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.Pdu.length of the respective Pdu). All other contained IPdus of a ContainerIPdu with static container layout have to be static length IPdus.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76543: [IpduM] Container Pdu not usable for CAN-FD

Problem description:

With the CAN-FD extension, the Container PDU concept has been introduced. Unfortunately this it not usable in a meaningful way for CAN-FD due to the 4-byte header for each contained PDU of a Container Pdu.

Agreed solution:

Proposed solution for the SWS IPduM:

Extend the enumeration of IpduMContainerHeaderSize by:
- IPDUM_HEADERTYPE_LONG
- IPDUM_HEADERTYPE_SHORT
- IPDUM_HEADERTYPE_NONE

The detailed proposed solution is contained here: https://svn.autosar.org/repos/work/09_WorkPackages/WP-A2/70_InternalDocuments/RfCs/76543-AUTOSAR_SWS_IPDUMultiplexer.doc

==========
Proposed solution TPS_SystemTemplate:

Extend the enumeration of ContainerIPduHeaderTypeEnum by:
- noHeader: No Header is used and the the location of each containedPdu in the ContainerPdu is statically configured.

Add the following optional attributes to ContainedIPduProps
- offset (PositiveInteger): Byte offset that describes the location of the ContainedPdu in the ContainerPdu if no header is used.
- updateIndicationBitPosition (PositiveInteger): The updateIndicationBit specifies the bit location of ContainedIPdu Update-Bit in the Container PDU. It indicates to the receivers that the ContainedIPdu in the ContainerIPdu was updated.

Add the following optional attributes to ContainerIPdu
- unusedBitPattern (PositiveInteger): IPduM fills not updated areas of the ContainerPdu with this byte-pattern.

Rewrite the Introduction of chapter 6.3.1 ContainerIPdu

For each IPdu which is put inside a ContainerIPdu, a header may be provided which determines which IPdu is contained (ContainedIPduProps.headerIdLongHeader or

headerIdShortHeader and what the size of that IPdu is (DLC during runtime). With this header mode the receivers are able to extract the individual contained IPdus again. As an alternative option to the usage of headers a statically configured layout of IPdus in the ContainerIPdu is supported.

Introduce the following constraints:

[constr_xxxx1] Mandatory offset if noHeader is used
For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu.containedPduTriggering with ContainerIPdu.headerType = noHeader the IPdu.containedIPduProps.offset shall be defined.

[constr_xxxx2] 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_xxxx3] 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_xxxx4] Only the last contained IPdu (according to the ContainedIPduProps.offset) of a ContainerIPdu with static container layout (i.e., a ContainerIPdu with ContainerIPduHeaderTypeEnum set to noHeader) may be a dynamic length PDU (i.e, a PDU that at runtime may exhibit a length different from the one statically configured via Pdu.length of the respective Pdu). All other contained PDUs of a ContainerIPdu with static container layout have to be static length PDUs.

Adapt the following spec items to:

[TPS_SYST_02098] Header id and header type of a contained IPdu
A contained IPdu shall always have the same headerId per header type (long or short header), regardless in which ContainerIPdu it is collected. If noHeader is set then the contained Ipdu does not need to have a headerId.

[TPS_SYST_02100] Relation between ContainerIPdu and contained IPdus on receiver side
On receiver side, it is not necessarily required to statically define which IPdus may be contained inside a ContainerIPdu if the header mode is used. Thus it would be possible to update the senders of ContainerIPdus and put different or additional IPdus inside.

Rework descriptions of the following classes and attributes in the model:

ContainerIPdu class description: Allows to collect several IPdus in one ContainerIPdu based on the headerType
ContainerIPdu.headerType attribute description: Defines whether and which header type is used (header id and length).

==========
Proposed solution SRS_IpduM:

- rename Chapter 6.1.4 to Dynamic I-PDU to Container Mapping
rename [SRS_IpduM_02820] to Dynamic I-PDU Mapping
add a dependency: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_0XXXX) but no mixture.
- add new Chapter Static I-PDU to Container Mapping
- add new SRS requirement: Static I-PDU Mapping
Type: Draft
Description: Multiple I-PDUs shall be mappable to a static position into a Container-PDU. If an I-PDU is mapped statically to a Container it is always transported in the same position of the Container-PDU.
Rationale: Support scenarios of Container Mapping where no header shall be used to reduce bandwidth in case always all contained PDU shall be sent.
Use Case: Efficient transmission of small PDUs on high bandwidth busses
Dependencies: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_02820) but no mixture.
–Last change on issue 76543 comment 100–

BW-C-Level:

Application	Specification	Bus
1	2	1

1.22 Specification Item constr_3406

Trace References:

none

Content:

In case that

- an ISignalIPdu is referenced by the SecuredIPdu with the SecuredIPdu.payload reference via the PduTriggering and
- the SecureCommunicationProps.authDataFreshnessStartPosition and SecureCommunicationProps.authDataFreshnessLength define the area in the ISignalIPdu that is taken to verify and generate the Freshness then

all ISignals that are mapped into the ISignalIPdu in front of the configured SecureCommunicationProps.authDataFreshnessStartPosition shall have a static length.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77336: [SECOC] Dynamic length PDUs (Container) not possible / clear

  Problem description:

  The SecOC SWS does not make any assumptions or restrictions about dynamic length PDUs. But with parameter SecOCAuthDataFreshnessStartPosition it could be impossible to really use dynamic length in SecOC.

  Additionally in SystemTemplate TPS the constraint constr_3139 talks only about some restrictions when dynamic length IPDUs are used. This would imply dynamic length IPDUs should be possible.

  Without dynamic length support it is also not possible to support securing complete IDPUM Containers.
  One possible solution could be (if dynamic length should not be supported by SecOC) to add a configuration option per IPDUM Container to send always maximum length (padding with 0) to have the static length again.
  If dynamic length shall be supported by SecOC a further problem will be the CAN-FD padding.

  Agreed solution:

  ==========
  AUTOSAR 4.3.1
  ==========

  SRS SecOC
  ----

  * Add new section next to 6.1.3.5 and new requirement [SRS_SecOC_xxxx1] Support of padding at lower layer modules and dynamic length Authentic I-PDUs.
  * Description: The SecOC module shall be applicable for the use cases with padding

at lower layer modules and with dynamic length Authentic I-PDUs.
* Rationale: At receiver side, received Secured I-PDU containing dynamic length Authentic I-PDU may also conatin padding bytes (added by lower layer modules of sender side, to fit to bus-specific L-PDU length constraints, e.g. CAN FD and FlexRay). In such case, receivers cannot identify number of bytes / byte position of the received payload.
* Use Case: dynamic length PDU on CAN FD and FlexRay
* Dependencies: [SRS_SecOC_00012]
* Supporting Material: -

Note: According to CM, RS Main should be referred. But upstream requirements in current (R4.3.0) SRS SecOC are from RS Features, and appropriate requirements are not available in RS Main.
(If we use RS Features, at least [RS_BRF_01568] [RS_BRF_01649] [RS_BRF_01712] [RS_BRF_01716] [RS_BRF_01752] [RS_BRF_02035] [RS_BRF_02036] [RS_BRF_02037] could be mapped to this requirements)

———————-
SWS SecOC
———————-

a1) Adapt Figure 4 in sec. 7.1.1.1

Change from
< (Figure of "Secured I-PDU = Authentic I-PDU | Freshness Value | Authenticator")
< Figure 4: Secured I-PDU contents
to
> (Figure of "Secured I-PDU = Secured I-PDU Header (optional) | Authentic I-PDU | Freshness Value (optional) | Authenticator")
> Figure 4: Secured I-PDU contents
==> to be done as RfC # 77807, not handled in this RfC.

a2) Add new requirements regarding the Secured I-PDU Header and related behavior

Add new requirement [SWS_SecOC_xxxx2] to define the Secured I-PDU Header
> The Secured I-PDU Header shall indicate the length of the Authentic I-PDU in bytes. The length of the Header shall be configurable by the parameter SecOCAuthPduHeaderLength.
> ()
> Note: the SecOC supports combined usage of authentication data in a separate

message (secured PDU collection) and Secured I-PDU Header. Also the SecOC covers dynamic length Authentic I-PDU.

Add new requirement [SWS_SecOC_xxxx3] for behavior at transmission (construction) of Secured I-PDUs
> For a Tx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall add the Secured I-PDU Header to the Secured I-PDU with the length of the Authentic I-PDU within the Secured I-PDU, to handle dynamic Authentic I-PDU.
> ()
> Note: Primary purpose of this Header is to indicate the position of Freshness Value and Authenticator in Secured I-PDUs with dynamic length Authentic I-PDU.
> Also some buses which cannot select arbitrary length of L-PDU (e.g. CAN FD and FlexRay) require this Header, because the position of Freshness Value and Authenticator is not always at the end of the Secured I-PDU, as lower layer modules (e.g. CanIf and FrIf) may add bus-specific padding bytes after processing at SecOC (then the L-PDU containing the Secured I-PDU with padding will be: Secured I-PDU = Secured I-PDU Header | Authentic I-PDU | Freshness Value | Authenticator | Bus-specific padding).

Add new requirement [SWS_SecOC_xxxx4] for behavior at reception of Secured I-PDUs
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall assume Secured I-PDU Header shall be available in the Secured I-PDU, to handle dynamic Authentic I-PDU.
> ()

Add new requirement [SWS_SecOC_xxxx5] for behavior at reception of Secured I-PDUs, the Header tells it's longer than the maximum length of the PDU
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0 and the length of Authentic I-PDU in the Header is longer than configured length (in case of dynamic length IPdus (containing a dynamical length signal), this value indicates the maximum data length) of the Authentic I-PDU, the SecOC module shall discard the I-PDU. In such case with SecOC_StartOfReception, BUFREQ_E_NOT_OK shall be returned (see [SWS_COMTYPE_00012]).
> ()
> Note: SecOC_RxIndication has no return value.

Add new requirement [SWS_SecOC_xxxx6] for behavior at reception of Se-

cured I-PDUs, the Header tells it's shorter than received I-PDU length (ignoring the padding at the end)
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall process Secured I-PDU Header, Authentic I-PDU (with the length specified by the Header), Freshness Value and Authenticator of the Rx Secured I-PDU. The rest of bytes in the Secured I-PDU shall be discarded.
> ()

a3) Add a configuration parameter to SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPdu, SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection/SecOCRxAuthenticPdu, SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPdu and SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPduCollection/SecOCTxAuthenticPdu to enable/disable Secured I-PDU Header per I-PDU
* SWS Item: ECUC_SecOC_xxxx1
* Name: SecOCAuthPduHeaderLength
* Description:
* This parameter indicates the length (in bytes) of the Secured I-PDU Header in the Secured I-PDU. The length of zero means there's no header in the PDU.
* Multiplicity: 0..1
* Range: 0..4
* Default: 0

a4) (removed)

a8) Update layout definition (construction) for Secured I-PDUs

Change [SWS_SecOC_00037]
from
< [SWS_SecOC_00037]
< The SecOC module shall construct the Secured I-PDU by adding the Freshness Value and the Authenticator to the Authentic I-PDU.
< (SRS_SecOC_00006)
< Note: The Freshness Counter and the Authenticator included as part of the Secured I-PDU may be truncated per configuration specific to the identifier of the Secured I-PDU. The scheme for the Secured I-PDU looks as follows:
< SecuredPDU = AuthenticIPDU | FreshnessValue [SecOCFreshnessValueTxLength] | Authenticator [SecOCAuthInfoTxLength]

to
> [SWS_SecOC_00037]
> The SecOC module shall construct the Secured I-PDU by adding the Secured I-PDU Header (optional), the Freshness Value (optional) and the Authenticator to the Authentic I-PDU.
> The scheme for the Secured I-PDU (includes the order in which the contents are structured in the Secured I-PDU) shall be compliant with below:
> SecuredPDU = SecuredIPDUHeader (optional) | AuthenticIPDU | FreshnessValue [SecOCFreshnessValueTxLength] (optional) | Authenticator [SecOCAuthInfoTxLength]
> (SRS_SecOC_00006)
> Note: The Freshness Counter and the Authenticator included as part of the Secured I-PDU may be truncated per configuration specific to the identifier of the Secured I-PDU. Also Freshness Value may be a part of Authentic I-PDU (see [SWS_SecOC_00219]).
==> to be done as RfC # 77807, not handled in this RfC.

a9) Add new constraints and notes after [SWS_SecOC_00219]:
> [constr_xxxx1] All signals before SecOCAuthDataFreshnessStartPosition within the Secured I-PDU shall have static length.
> Note: SecOC can use a part of the Authentic I-PDU as freshness when SecOCUseAuthDataFreshness=true, only if the part of the Authentic I-PDU to be used as the freshness is always available at same position in the Authentic I-PDU.

> [constr_xxxx2] Any container I-PDU which contains multiple contained I-PDUs shall be set SecOCUseAuthDataFreshness=false.
> Note: For container PDUs, normally it cannot be ensured which PDU will be put in which position (depends on various timing and trigger conditions). Therefore, container I-PDUs with multiple contained I-PDUs cannot have FV within the Authentic I-PDU.

a10) Adapt Figure 5
==> to be done as RfC # 77807, not handled in this RfC.

a11) Adapt SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection/SecOCRxAuthenticPdu and SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPduCollection/SecOCTxAuthenticPdu

Change the description of SecOCRxAuthenticPdu (ECUC_SecOC_00061)

from
< This container specifies the Authetic Pdu that is received by the SecOC module from the PduR.
to
> This container specifies the PDU (that is received by the SecOC module from the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.

Change the description of SecOCTxAuthenticPdu ECUC_SecOC_00072
from
< This container specifies the Authetic Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated.
to
> This container specifies the PDU (that is transmitted by the SecOC module to the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.

———————-
TPS System Template (SysT)
———————-

a5) Add new attribute to SecuredIPdu (Table 6.46) which enables SecOCAuthPduHeaderLength>0
* Attribute: useSecuredPduHeader
* Type: SecuredPduHeaderEnum
* Mul.: 0..1
* Kind: attr
* Desc: This attribute defines the size of the header which is inserted into the SecuredIPdu. If this attribute is set to anything but noHeader, the SecuredIPdu contains the Secured I-PDU Header to indicate the length of the AuthenticIPdu. The AuthenticIPdu contains the original payload, i.e. the secured data.

SecuredPduHeaderEnum
- noHeader
- securedPduHeader08Bit
- securedPduHeader16Bit
- securedPduHeader32Bit
Desc: Defines the header which will be inserted into the SecuredIPdu.

a6) Change the description of IPduPort.rxSecurityVerification in Table 6.3: IPduPort:
This attribute defines the bypassing of signature authentication or MAC verification in the receiving ECU.
If not defined or set to true the signature authentication or MAC verification shall be performed for the SecuredIPdu.

If set to false the signature authentication or MAC verification shall not be performed for the SecuredIPdu.

Removed [constr_3139].

TPS_SysT_xxxx2: Setting of useSecuredPduHeader attribute
The useSecuredPduHeader shall be set to a value other than noHeader if the length of the payload Pdu is dynamic and is transmitted over a network which may insert padding bytes depending on the length (e.g. CANFD, Flexray).

Add a note below TPS_SysT_xxxx2:

Please note that the dynamic-length Pdu can be an ISignalIPdu that contains a SystemSignal with dynamicLength set to true. In general it is not possible to run diagnostics on fixed-length Pdus. Therefore, there is a probability that at least a subset of DcmIPdus and UserDefinedIPdus can have dynamic length.

a7) Add upstream mapping between useSecuredPduHeader (SysT) and SecOCAuthPduHeaderLength (EcuC) in C.1.4 SecOc Mapping

b4) Add upstream mapping between rxSecurityVerification (SysT) and SecOCSecuredRxPduVerification (EcuC) in C.1.4 SecOc Mapping
Mapping rule: SecOCSecuredRxPduVerification is True if rxSecurityVerification is not defined, otherwise SecOCSecuredRxPduVerification = rxSecurityVerification

———————-
SRS SecOC
———————-

* Add new section next to 6.1.3.5 (or 6.2.1.1) and new requirement [SRS_SecOC_xxxx2] Support of capability to extract Authentic I-PDU without Authentication
* Description: The SecOC module shall be capable to extract Authentic I-PDU from Secured I-PDU, without Authentication.
* Rationale: SecOC can be used as an extractor of Authentic I-PDU from Secured I-PDU, to enable low latency GW behavior when a part of downstream communication clusters doesn't require authentication of PDUs.
* Use Case: Gateway
* Dependencies: [SRS_SecOC_00025]
* Supporting Material: -

Note: According to CM, RS Main should be referred. But upstream requirments in

current (R4.3.0) SRS SecOC are from RS Features, and appropriate requirements are not available in RS Main.
(If we use RS Features, at least [RS_BRF_02035] [RS_BRF_02036] [RS_BRF_02037] could be mapped to this requirements)

———————-
SWS SecOC
———————-

b1) Remove [constr_3139] (not [constr_3193] – sorry, constr_3193 is typo in my comment # 10)

b2) Add new requirements regarding skipped authentication behavior at SecOC (just remove FV/MAC from Secured I-PDU)

* Add new section "Extracting Authentic I-PDU without Authentication at SecOC" or "Skipping Authentication for Secured I-PDUs at SecOC"
* Add new requirement [SWS_SecOC_xxxx7] for behavior of SecOC at reception of Secured I-PDUs without Authentication
> For a Rx Secured I-PDU with SecOCSecuredRxPduVerification=false, the SecOC module shall extract the Authentic I-PDU using the length specified by the Secured I-PDU Header without Authentication.
> ()

b3) Add a configuration parameter to SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPdu and SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection to control authentication behavior at SecOC

* SWS Item: ECUC_SecOC_xxxx3
* Name: SecOCSecuredRxPduVerification
* Description: This parameter defines whether the signature authentication or MAC verification shall be performed on this Secured I-PDU. If set to false, the SecOC module extracts the Authentic I-PDU from the Secured I-PDU without verification.
* Multiplicity: 1
* Type: EcucBooleanParamDef
* Default value: false
* Post-Build Variant Value: true
* Value Configuration Class:
* Pre-compile time: X All Variants

* Scope / Dependency: scope: local
–Last change on issue 77336 comment 69–

BW-C-Level:

Application	Specification	Bus
1	4	4

1.23 Specification Item constr_3407

Trace References:

none

Content:

If a ContainerIPdu that is referenced by the SecuredIPdu with the SecuredIPdu.payload reference via the PduTriggering contains a dynamic layout (i.e. ContainerIPdu.Container IPdu.headerType is set to ContainerIPduHeaderTypeEnum.longHeader or ContainerIPdu HeaderTypeEnum.shortHeader) and multiple contained IPdus then each IPduPort that is referenced by the PduTriggering of the SecuredIPdu shall have the attribute IPduPort.use AuthDataFreshness set to false.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77336: [SECOC] Dynamic length PDUs (Container) not possible / clear

  Problem description:

  The SecOC SWS does not make any assumptions or restrictions about dynamic length PDUs. But with parameter SecOCAuthDataFreshnessStartPosition it could be impossible to really use dynamic length in SecOC.

  Additionally in SystemTemplate TPS the constraint constr_3139 talks only about some restrictions when dynamic length IPDUs are used. This would imply dynamic length IPDUs should be possible.

  Without dynamic length support it is also not possible to support securing complete IDPUM Containers.
  One possible solution could be (if dynamic length should not be supported by SecOC) to add a configuration option per IPDUM Container to send always maximum length (padding with 0) to have the static length again.
  If dynamic length shall be supported by SecOC a further problem will be the CAN-FD padding.

Agreed solution:

==========
AUTOSAR 4.3.1
==========

_____-
SRS SecOC
_____-

* Add new section next to 6.1.3.5 and new requirement [SRS_SecOC_xxxx1] Support of padding at lower layer modules and dynamic length Authentic I-PDUs.
* Description: The SecOC module shall be applicable for the use cases with padding at lower layer modules and with dynamic length Authentic I-PDUs.
* Rationale: At receiver side, received Secured I-PDU containing dynamic length Authentic I-PDU may also conatin padding bytes (added by lower layer modules of sender side, to fit to bus-specific L-PDU length constraints, e.g. CAN FD and FlexRay). In such case, receivers cannot identify number of bytes / byte position of the received payload.
* Use Case: dynamic length PDU on CAN FD and FlexRay
* Dependencies: [SRS_SecOC_00012]
* Supporting Material: -

Note: According to CM, RS Main should be referred. But upstream requirements in current (R4.3.0) SRS SecOC are from RS Features, and appropriate requirements are not available in RS Main.
(If we use RS Features, at least [RS_BRF_01568] [RS_BRF_01649] [RS_BRF_01712] [RS_BRF_01716] [RS_BRF_01752] [RS_BRF_02035] [RS_BRF_02036] [RS_BRF_02037] could be mapped to this requirements)

_____-
SWS SecOC
_____-

a1) Adapt Figure 4 in sec. 7.1.1.1

Change from
< (Figure of "Secured I-PDU = Authentic I-PDU | Freshness Value | Authenticator")
< Figure 4: Secured I-PDU contents
to
> (Figure of "Secured I-PDU = Secured I-PDU Header (optional) | Authentic I-PDU |

Freshness Value (optional) | Authenticator")
> Figure 4: Secured I-PDU contents
==> to be done as RfC # 77807, not handled in this RfC.

a2) Add new requirements regarding the Secured I-PDU Header and related behavior

Add new requirement [SWS_SecOC_xxxx2] to define the Secured I-PDU Header
> The Secured I-PDU Header shall indicate the length of the Authentic I-PDU in bytes. The length of the Header shall be configurable by the parameter SecOCAuthPduHeaderLength.
> ()
> Note: the SecOC supports combined usage of authentication data in a separate message (secured PDU collection) and Secured I-PDU Header. Also the SecOC covers dynamic length Authentic I-PDU.

Add new requirement [SWS_SecOC_xxxx3] for behavior at transmission (construction) of Secured I-PDUs
> For a Tx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall add the Secured I-PDU Header to the Secured I-PDU with the length of the Authentic I-PDU within the Secured I-PDU, to handle dynamic Authentic I-PDU.
> ()
> Note: Primary purpose of this Header is to indicate the position of Freshness Value and Authenticator in Secured I-PDUs with dynamic length Authentic I-PDU.
> Also some buses which cannot select arbitrary length of L-PDU (e.g. CAN FD and FlexRay) require this Header, because the position of Freshness Value and Authenticator is not always at the end of the Secured I-PDU, as lower layer modules (e.g. CanIf and FrIf) may add bus-specific padding bytes after processing at SecOC (then the L-PDU containing the Secured I-PDU with padding will be: Secured I-PDU = Secured I-PDU Header | Authentic I-PDU | Freshness Value | Authenticator | Bus-specific padding).

Add new requirement [SWS_SecOC_xxxx4] for behavior at reception of Secured I-PDUs
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall assume Secured I-PDU Header shall be available in the Secured I-PDU, to handle dynamic Authentic I-PDU.
> ()

Add new requirement [SWS_SecOC_xxxx5] for behavior at reception of Secured I-PDUs, the Header tells it's longer than the maximum length of the PDU
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0 and the length of Authentic I-PDU in the Header is longer than configured length (in case of dynamic length IPdus (containing a dynamical length signal), this value indicates the maximum data length) of the Authentic I-PDU, the SecOC module shall discard the I-PDU. In such case with SecOC_StartOfReception, BUFREQ_E_NOT_OK shall be returned (see [SWS_COMTYPE_00012]).
> ()
> Note: SecOC_RxIndication has no return value.

Add new requirement [SWS_SecOC_xxxx6] for behavior at reception of Secured I-PDUs, the Header tells it's shorter than received I-PDU length (ignoring the padding at the end)
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall process Secured I-PDU Header, Authentic I-PDU (with the length specified by the Header), Freshness Value and Authenticator of the Rx Secured I-PDU. The rest of bytes in the Secured I-PDU shall be discarded.
> ()

a3) Add a configuration parameter to SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPdu, SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection/SecOCRxAuthenticPdu, SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPdu and SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPduCollection/SecOCTxAuthenticPdu to enable/disable Secured I-PDU Header per I-PDU
* SWS Item: ECUC_SecOC_xxxx1
* Name: SecOCAuthPduHeaderLength
* Description:
* This parameter indicates the length (in bytes) of the Secured I-PDU Header in the Secured I-PDU. The length of zero means there's no header in the PDU.
* Multiplicity: 0..1
* Range: 0..4
* Default: 0

a4) (removed)

a8) Update layout definition (construction) for Secured I-PDUs

Change [SWS_SecOC_00037]
from
< [SWS_SecOC_00037]
< The SecOC module shall construct the Secured I-PDU by adding the Freshness Value and the Authenticator to the Authentic I-PDU.
< (SRS_SecOC_00006)
< Note: The Freshness Counter and the Authenticator included as part of the Secured I-PDU may be truncated per configuration specific to the identifier of the Secured I-PDU. The scheme for the Secured I-PDU looks as follows:
< SecuredPDU = AuthenticIPDU | FreshnessValue [SecOCFreshnessValueTxLength] | Authenticator [SecOCAuthInfoTxLength]
to
> [SWS_SecOC_00037]
> The SecOC module shall construct the Secured I-PDU by adding the Secured I-PDU Header (optional), the Freshness Value (optional) and the Authenticator to the Authentic I-PDU.
> The scheme for the Secured I-PDU (includes the order in which the contents are structured in the Secured I-PDU) shall be compliant with below:
> SecuredPDU = SecuredIPDUHeader (optional) | AuthenticIPDU | FreshnessValue [SecOCFreshnessValueTxLength] (optional) | Authenticator [SecOCAuthInfoTxLength]
> (SRS_SecOC_00006)
> Note: The Freshness Counter and the Authenticator included as part of the Secured I-PDU may be truncated per configuration specific to the identifier of the Secured I-PDU. Also Freshness Value may be a part of Authentic I-PDU (see [SWS_SecOC_00219]).
==> to be done as RfC # 77807, not handled in this RfC.

a9) Add new constraints and notes after [SWS_SecOC_00219]:
> [constr_xxxx1] All signals before SecOCAuthDataFreshnessStartPosition within the Secured I-PDU shall have static length.
> Note: SecOC can use a part of the Authentic I-PDU as freshness when SecOCUseAuthDataFreshness=true, only if the part of the Authentic I-PDU to be used as the freshness is always available at same position in the Authentic I-PDU.

> [constr_xxxx2] Any container I-PDU which contains multiple contained I-PDUs shall be set SecOCUseAuthDataFreshness=false.
> Note: For container PDUs, normally it cannot be ensured which PDU will be put in which position (depends on various timing and trigger conditions). Therefore, con-

tainer I-PDUs with multiple contained I-PDUs cannot have FV within the Authentic I-PDU.

a10) Adapt Figure 5
==> to be done as RfC # 77807, not handled in this RfC.

a11) Adapt SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection/SecOCRxAuthenticPdu and SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPduCollection/SecOCTxAuthenticPdu

Change the description of SecOCRxAuthenticPdu (ECUC_SecOC_00061)
from
< This container specifies the Authetic Pdu that is received by the SecOC module from the PduR.
to
> This container specifies the PDU (that is received by the SecOC module from the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.

Change the description of SecOCTxAuthenticPdu ECUC_SecOC_00072
from
< This container specifies the Authetic Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated.
to
> This container specifies the PDU (that is transmitted by the SecOC module to the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.

———————-
TPS System Template (SysT)
———————-

a5) Add new attribute to SecuredIPdu (Table 6.46) which enables SecOCAuthPduHeaderLength>0
* Attribute: useSecuredPduHeader
* Type: SecuredPduHeaderEnum
* Mul.: 0..1
* Kind: attr
* Desc: This attribute defines the size of the header which is inserted into the SecuredIPdu. If this attribute is set to anything but noHeader, the SecuredIPdu contains the Secured I-PDU Header to indicate the length of the AuthenticIPdu. The

AuthenticIPdu contains the original payload, i.e. the secured data.

SecuredPduHeaderEnum
- noHeader
- securedPduHeader08Bit
- securedPduHeader16Bit
- securedPduHeader32Bit
Desc: Defines the header which will be inserted into the SecuredIPdu.

a6) Change the description of IPduPort.rxSecurityVerification in Table 6.3: IPduPort:
This attribute defines the bypassing of signature authentication or MAC verification in the receiving ECU.
If not defined or set to true the signature authentication or MAC verification shall be performed for the SecuredIPdu.
If set to false the signature authentication or MAC verification shall not be performed for the SecuredIPdu.

Removed [constr_3139].

TPS_SysT_xxxx2: Setting of useSecuredPduHeader attribute
The useSecuredPduHeader shall be set to a value other than noHeader if the length of the payload Pdu is dynamic and is transmitted over a network which may insert padding bytes depending on the length (e.g. CANFD, Flexray).

Add a note below TPS_SysT_xxxx2:

Please note that the dynamic-length Pdu can be an ISignalIPdu that contains a SystemSignal with dynamicLength set to true. In general it is not possible to run diagnostics on fixed-length Pdus. Therefore, there is a probability that at least a subset of DcmIPdus and UserDefinedIPdus can have dynamic length.

a7) Add upstream mapping between useSecuredPduHeader (SysT) and SecOCAuthPduHeaderLength (EcuC) in C.1.4 SecOc Mapping

b4) Add upstream mapping between rxSecurityVerification (SysT) and SecOCSecuredRxPduVerification (EcuC) in C.1.4 SecOc Mapping
Mapping rule: SecOCSecuredRxPduVerification is True if rxSecurityVerification is not defined, otherwise SecOCSecuredRxPduVerification = rxSecurityVerification

———————-
SRS SecOC
———————-

* Add new section next to 6.1.3.5 (or 6.2.1.1) and new requirement [SRS_SecOC_xxxx2] Support of capability to extract Authentic I-PDU without Authentication
* Description: The SecOC module shall be capable to extract Authentic I-PDU from Secured I-PDU, without Authentication.
* Rationale: SecOC can be used as an extractor of Authentic I-PDU from Secured I-PDU, to enable low latency GW behavior when a part of downstream communication clusters doesn't require authentication of PDUs.
* Use Case: Gateway
* Dependencies: [SRS_SecOC_00025]
* Supporting Material: -

Note: According to CM, RS Main should be referred. But upstream requirments in current (R4.3.0) SRS SecOC are from RS Features, and appropriate requirements are not available in RS Main.
(If we use RS Features, at least [RS_BRF_02035] [RS_BRF_02036] [RS_BRF_02037] could be mapped to this requirements)

———————-
SWS SecOC
———————-

b1) Remove [constr_3139] (not [constr_3193] – sorry, constr_3193 is typo in my comment # 10)

b2) Add new requirements regarding skipped authentication behavior at SecOC (just remove FV/MAC from Secured I-PDU)

* Add new section "Extracting Authentic I-PDU without Authentication at SecOC" or "Skipping Authentication for Secured I-PDUs at SecOC"
* Add new requirement [SWS_SecOC_xxxx7] for behavior of SecOC at reception of Secured I-PDUs without Authentication
> For a Rx Secured I-PDU with SecOCSecuredRxPduVerification=false, the SecOC module shall extract the Authentic I-PDU using the length specified by the Secured I-PDU Header without Authentication.
> ()

b3) Add a configuration parameter to SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPdu and SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection to control

authentication behavior at SecOC

* SWS Item: ECUC_SecOC_xxxx3
* Name: SecOCSecuredRxPduVerification
* Description: This parameter defines whether the signature authentication or MAC verification shall be performed on this Secured I-PDU. If set to false, the SecOC module extracts the Authentic I-PDU from the Secured I-PDU without verification.
* Multiplicity: 1
* Type: EcucBooleanParamDef
* Default value: false
* Post-Build Variant Value: true
* Value Configuration Class:
* Pre-compile time: X All Variants
* Scope / Dependency: scope: local
–Last change on issue 77336 comment 69–

BW-C-Level:

Application	Specification	Bus
1	4	4

1.24 Specification Item TPS_SYST_01120

Trace References:

none

Content:

If a dedicated ISignalMapping for at least one ISignal within an ISignalGroup exists the implicit mapping on the basis of Referrable.shortNames is no longer applicable for any ISignal within that ISignalGroup.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74600: Signal Mapping by shortName

  Problem description:

  There's a not in the document that breaks:

  "By default, identical shortNames of ISignal elements identify correlating ISignals within the respective ISignalGroups referenced in the scope of ISignalMapping."

  I'm not sure what the background for this note is and whether it should be

made a specification item (especially as the shortName-based routing is mentioned in TPS_Syst_01120.

On the other hand: the model allows for a dedicated mapping of routing relations. Why is it necessary to have a name-based, implicit routing relation?

Agreed solution:

Change the note behind constr_3052 to a new specification item:

[TPS_SYST_xxxx1] Routing of ISignals of ISignalGroups
When performing a signal group routing two approaches are supported for the pairing of the included ISignals:
- implicit mapping: the ISignalMapping points in the sourceSignal role to an ISignalTriggering of an ISignalGroup and no ISignalMappings are defined for the included ISignals. Identical shortNames of ISignal elements identify correlating ISignals between the source and the target in the scope of the ISignalMapping.
- explicit mapping: the ISignalMapping points in the sourceSignal role to an ISignalTriggering of an ISignalGroup and in addition explicitly specified ISignalMappings define which ISignals correlate to each other.

Change [TPS_SYST_01120] to (add implicit mapping):
[TPS_SYST_01120] Precedence of ISignalMappings
If a dedicated ISignalMapping for at least one ISignal within an ISignalGroup exists the implicit mapping on the basis of shortNames is no longer applicable for any ISignal within that ISignalGroup.

Reference in [constr_3053] to the new spec item:
[constr_3053] Complete ISignalMapping of target ISignalGroup
If an ISignalGroup is referenced by a targetSignal then [TPS_SYST_01120] applies for each of the contained ISignal of that ISignalGroup.
–Last change on issue 74600 comment 8–

BW-C-Level:

Application	Specification	Bus
1	3	1

1.25 Specification Item TPS_SYST_02005

Trace References:

none

Content:

Low-level routing of J1939DcmIPdus : In case of J1939DcmIPdus the Pdus are handled like IPdus and the PduTriggering and IPduPort shall be defined.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #78380: [SysT] Low level routing of J1939DcmIPdus is not possible

  Problem description:

  TPS_SYST_02005 describes "Low-level routing of J1939DcmIPdus". J1939DcmIPdus can have more than 8 bytes of data, then a low-level routing of NPdus is required, or they have up to 8 bytes of data, then they can be routed directly. A low-level routing of the J1939DcmIPdus is never required/possible.

  Agreed solution:

  Remove [TPS_SYST_02005]
  –Last change on issue 78380 comment 2–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.26 Specification Item TPS_SYST_02098

Trace References:

RS_SYST_00055

Content:

A contained IPdu shall always have the same headerId per header type (long or short header), regardless in which ContainerIPdu it is collected. If ContainerIPduHeaderType Enum.noHeader is set then the contained IPdu does not need to have a headerId.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76543: [IpduM] Container Pdu not usable for CAN-FD

  Problem description:

  With the CAN-FD extension, the Container PDU concept has been introduced. Unfortunately this it not usable in a meaningful way for CAN-FD due to the 4-byte header for each contained PDU of a Container Pdu.

Agreed solution:

Proposed solution for the SWS IPduM:

Extend the enumeration of IpduMContainerHeaderSize by:
- IPDUM_HEADERTYPE_LONG
- IPDUM_HEADERTYPE_SHORT
- IPDUM_HEADERTYPE_NONE

The detailed proposed solution is contained here: https://svn.autosar.org/repos/work/09_WorkPackages/WP-A2/70_InternalDocuments/RfCs/76543-AUTOSAR_SWS_IPDUMultiplexer.doc

==========
Proposed solution TPS_SystemTemplate:

Extend the enumeration of ContainerIPduHeaderTypeEnum by:
- noHeader: No Header is used and the the location of each containedPdu in the ContainerPdu is statically configured.

Add the following optional attributes to ContainedIPduProps
- offset (PositiveInteger): Byte offset that describes the location of the ContainedPdu in the ContainerPdu if no header is used.
- updateIndicationBitPosition (PositiveInteger): The updateIndicationBit specifies the bit location of ContainedIPdu Update-Bit in the Container PDU. It indicates to the receivers that the ContainedIPdu in the ContainerIPdu was updated.

Add the following optional attributes to ContainerIPdu
- unusedBitPattern (PositiveInteger): IPduM fills not updated areas of the ContainerPdu with this byte-pattern.

Rewrite the Introduction of chapter 6.3.1 ContainerIPdu

For each IPdu which is put inside a ContainerIPdu, a header may be provided which determines which IPdu is contained (ContainedIPduProps.headerIdLongHeader or headerIdShortHeader and what the size of that IPdu is (DLC during runtime). With this header mode the receivers are able to extract the individual contained IPdus again. As an alternative option to the usage of headers a statically configured layout of IPdus in the ContainerIPdu is supported.

Introduce the following constraints:

[constr_xxxx1] Mandatory offset if noHeader is used
For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu.containedPduTriggering with ContainerIPdu.headerType = noHeader the IPdu.containedIPduProps.offset shall be defined.

[constr_xxxx2] 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_xxxx3] 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_xxxx4] Only the last contained IPdu (according to the ContainedIPduProps.offset) of a ContainerIPdu with static container layout (i.e., a ContainerIPdu with ContainerIPduHeaderTypeEnum set to noHeader) may be a dynamic length PDU (i.e, a PDU that at runtime may exhibit a length different from the one statically configured via Pdu.length of the respective Pdu). All other contained PDUs of a ContainerIPdu with static container layout have to be static length PDUs.

Adapt the following spec items to:

[TPS_SYST_02098] Header id and header type of a contained IPdu
A contained IPdu shall always have the same headerId per header type (long or short header), regardless in which ContainerIPdu it is collected. If noHeader is set then the contained Ipdu does not need to have a headerId.

[TPS_SYST_02100] Relation between ContainerIPdu and contained IPdus on receiver side
On receiver side, it is not necessarily required to statically define which IPdus may be contained inside a ContainerIPdu if the header mode is used. Thus it would be possible to update the senders of ContainerIPdus and put different or additional IPdus inside.

Rework descriptions of the following classes and attributes in the model:

ContainerIPdu class description: Allows to collect several IPdus in one Con-

tainerIPdu based on the headerType
ContainerIPdu.headerType attribute description: Defines whether and which header type is used (header id and length).

==========
Proposed solution SRS_IpduM:

- rename Chapter 6.1.4 to Dynamic I-PDU to Container Mapping
rename [SRS_IpduM_02820] to Dynamic I-PDU Mapping
add a dependency: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_0XXXX) but no mixture.
- add new Chapter Static I-PDU to Container Mapping
- add new SRS requirement: Static I-PDU Mapping
Type: Draft
Description: Multiple I-PDUs shall be mappable to a static position into a Container-PDU. If an I-PDU is mapped statically to a Container it is always transported in the same position of the Container-PDU.
Rationale: Support scenarios of Container Mapping where no header shall be used to reduce bandwidth in case always all contained PDU shall be sent.
Use Case: Efficient transmission of small PDUs on high bandwidth busses
Dependencies: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_02820) but no mixture.
–Last change on issue 76543 comment 100–

BW-C-Level:

Application	Specification	Bus
1	2	1

1.27 Specification Item TPS_SYST_02100

Trace References:

RS_SYST_00055

Content:

On receiver side, it is not necessarily required to statically define which IPdus may be contained inside a ContainerIPdu if the header mode is used. Thus it would be possible to update the senders of ContainerIPdus and put different or additional IPdus inside.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76543: [IpduM] Container Pdu not usable for CAN-FD

  Problem description:

  With the CAN-FD extension, the Container PDU concept has been introduced. Unfortunately this it not usable in a meaningful way for CAN-FD due to the 4-byte header for each contained PDU of a Container Pdu.

  Agreed solution:

  Proposed solution for the SWS IPduM:

  Extend the enumeration of IpduMContainerHeaderSize by:
  - IPDUM_HEADERTYPE_LONG
  - IPDUM_HEADERTYPE_SHORT
  - IPDUM_HEADERTYPE_NONE

  The detailed proposed solution is contained here: https://svn.autosar.org/repos/work/09_WorkPackages/WP-A2/70_InternalDocuments/RfCs/76543-AUTOSAR_SWS_IPDUMultiplexer.doc

  ==========
  Proposed solution TPS_SystemTemplate:

  Extend the enumeration of ContainerIPduHeaderTypeEnum by:
  - noHeader: No Header is used and the the location of each containedPdu in the ContainerPdu is statically configured.

  Add the following optional attributes to ContainedIPduProps
  - offset (PositiveInteger): Byte offset that describes the location of the ContainedPdu in the ContainerPdu if no header is used.
  - updateIndicationBitPosition (PositiveInteger): The updateIndicationBit specifies the bit location of ContainedIPdu Update-Bit in the Container PDU. It indicates to the receivers that the ContainedIPdu in the ContainerIPdu was updated.

  Add the following optional attributes to ContainerIPdu
  - unusedBitPattern (PositiveInteger): IPduM fills not updated areas of the ContainerPdu with this byte-pattern.

  Rewrite the Introduction of chapter 6.3.1 ContainerIPdu

  For each IPdu which is put inside a ContainerIPdu, a header may be provided which

determines which IPdu is contained (ContainedIPduProps.headerIdLongHeader or headerIdShortHeader and what the size of that IPdu is (DLC during runtime). With this header mode the receivers are able to extract the individual contained IPdus again. As an alternative option to the usage of headers a statically configured layout of IPdus in the ContainerIPdu is supported.

Introduce the following constraints:

[constr_xxxx1] Mandatory offset if noHeader is used
For each IPdu which is assigned to a ContainerIPdu in the role ContainerIPdu.containedPduTriggering with ContainerIPdu.headerType = noHeader the IPdu.containedIPduProps.offset shall be defined.

[constr_xxxx2] 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_xxxx3] 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_xxxx4] Only the last contained IPdu (according to the ContainedIPduProps.offset) of a ContainerIPdu with static container layout (i.e., a ContainerIPdu with ContainerIPduHeaderTypeEnum set to noHeader) may be a dynamic length PDU (i.e, a PDU that at runtime may exhibit a length different from the one statically configured via Pdu.length of the respective Pdu). All other contained PDUs of a ContainerIPdu with static container layout have to be static length PDUs.

Adapt the following spec items to:

[TPS_SYST_02098] Header id and header type of a contained IPdu
A contained IPdu shall always have the same headerId per header type (long or short header), regardless in which ContainerIPdu it is collected. If noHeader is set then the contained Ipdu does not need to have a headerId.

[TPS_SYST_02100] Relation between ContainerIPdu and contained IPdus on receiver side
On receiver side, it is not necessarily required to statically define which IPdus may be contained inside a ContainerIPdu if the header mode is used. Thus it would be possible to update the senders of ContainerIPdus and put different or additional

IPdus inside.

Rework descriptions of the following classes and attributes in the model:

ContainerIPdu class description: Allows to collect several IPdus in one ContainerIPdu based on the headerType
ContainerIPdu.headerType attribute description: Defines whether and which header type is used (header id and length).

==========
Proposed solution SRS_IpduM:

- rename Chapter 6.1.4 to Dynamic I-PDU to Container Mapping
rename [SRS_IpduM_02820] to Dynamic I-PDU Mapping
add a dependency: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_0XXXX) but no mixture.
- add new Chapter Static I-PDU to Container Mapping
- add new SRS requirement: Static I-PDU Mapping
Type: Draft
Description: Multiple I-PDUs shall be mappable to a static position into a Container-PDU. If an I-PDU is mapped statically to a Container it is always transported in the same position of the Container-PDU.
Rationale: Support scenarios of Container Mapping where no header shall be used to reduce bandwidth in case always all contained PDU shall be sent.
Use Case: Efficient transmission of small PDUs on high bandwidth busses
Dependencies: A Container-PDU shall either support static or dynamic mapping (see SRS_IpduM_02820) but no mixture.
–Last change on issue 76543 comment 100–

BW-C-Level:

Application	Specification	Bus
1	2	1

1.28 Specification Item TPS_SYST_02112

Trace References:

none

Content:

EventHandler.EventHandler.applicationEndpoint shall only be used if the EventHandler is provided on a different ApplicationEndpoint (TP Port) that than the aggregating Application Endpoint that contains the ProvidedServiceInstance. In all other cases this reference can be ignored.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76972: Typo in [TPS_SYST_02112] Usage of EventHandler.applicationEndpoint reference

  Problem description:

  _____

  Name: Robert Sakretz
  Phone:
  Role: WP-M

  _____

  Description/Motivation:

  [TPS_SYST_02112] Usage of EventHandler.applicationEndpoint reference
  EventHandler.applicationEndpoint shall only be used if the EventHandler is provided on a different ApplicationEndpoint (TP Port) that the aggregating ApplicationEndpoint that contains the ProvidedServiceInstance. In all other cases this reference can be ignored.

  "... is provided on a different ApplicationEndpoint (TP Port) _that_ the aggregating ApplicationEndpoint..."

  should be _than_

  Agreed solution:

  Update TPS_SYST_02112 to:

  EventHandler.applicationEndpoint shall only be used if the EventHandler is provided on a different ApplicationEndpoint (TP Port) than the aggregating ApplicationEndpoint that contains the ProvidedServiceInstance. In all other cases this reference can be ignored.

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.29 Specification Item TPS_SYST_02160

Trace References:

none

Content:

A mix of EthernetPhysicalChannels with different Identifiable.category values within an EthernetCluster is currently not supported by AUTOSAR.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76287: TPS_SYST_02160 cannot be implemented

  Problem description:

  TPS_SYST_02160 states a negative requirement:

  EthernetPhysicalChannels with different category values
  are not allowed within an EthernetCluster d A mix of EthernetPhysicalChannels
  with different category values within an EthernetCluster is currently
  not supported by AUTOSAR. c()

  This cannot be implemented. It can only be checked by a tool and therefore needs be converted into a constraint.

  Agreed solution:

  Change Spec Items TPS_SYST_02160 to a new constraint:

  [constr_xxxxx] EthernetPhysicalChannels with different category values
  are not allowed within an EthernetCluster d A mix of EthernetPhysicalChannels
  with different category values within an EthernetCluster is currently
  not supported by AUTOSAR. c()
  –Last change on issue 76287 comment 4–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.30 Specification Item TPS_SYST_02162

Trace References:

none

Content:

When performing a signal group routing two approaches are supported for the pairing of the included ISignals:

- implicit mapping: the ISignalMapping points in the ISignalMapping.sourceSignal role to an ISignalTriggering of an ISignalGroup and no ISignalMappings are defined for the included ISignals. Identical Referrable.shortNames of ISignal elements identify correlating ISignals between the source and the target in the scope of the ISignal Mapping.
- explicit mapping: the ISignalMapping points in the ISignalMapping.sourceSignal role to an ISignalTriggering of an ISignalGroup and in addition explicitly specified ISignal Mappings define which ISignals correlate to each other.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74600: Signal Mapping by shortName

  Problem description:

  There's a not in the document that breaks:

  "By default, identical shortNames of ISignal elements identify correlating ISignals within the respective ISignalGroups referenced in the scope of ISignalMapping."

  I'm not sure what the background for this note is and whether it should be made a specification item (especially as the shortName-based routing is mentioned in TPS_Syst_01120.

  On the other hand: the model allows for a dedicated mapping of routing relations. Why is it necessary to have a name-based, implicit routing relation?

  Agreed solution:

  Change the note behind constr_3052 to a new specification item:

  [TPS_SYST_xxxx1] Routing of ISignals of ISignalGroups
  When performing a signal group routing two approaches are supported for the pairing of the included ISignals:
  - implicit mapping: the ISignalMapping points in the sourceSignal role to an ISignalTriggering of an ISignalGroup and no ISignalMappings are defined for the included ISignals. Identical shortNames of ISignal elements identify correlating ISignals between the source and the target in the scope of the ISignalMapping.
  - explicit mapping: the ISignalMapping points in the sourceSignal role to an ISignal-

Triggering of an ISignalGroup and in addition explicitly specified ISignalMappings define which ISignals correlate to each other.

Change [TPS_SYST_01120] to (add implicit mapping):
[TPS_SYST_01120] Precedence of ISignalMappings
If a dedicated ISignalMapping for at least one ISignal within an ISignalGroup exists the implicit mapping on the basis of shortNames is no longer applicable for any ISignal within that ISignalGroup.

Reference in [constr_3053] to the new spec item:
[constr_3053] Complete ISignalMapping of target ISignalGroup
If an ISignalGroup is referenced by a targetSignal then [TPS_SYST_01120] applies for each of the contained ISignal of that ISignalGroup.
–Last change on issue 74600 comment 8–

BW-C-Level:

Application	Specification	Bus
1	3	1

1.31 Specification Item TPS_SYST_02163

Trace References:

none

Content:

GlobalTimeDomain.GlobalTimeDomain.syncLossTimeout shall be specified for Global TimeDomains that have an aggregated slave and for all other cases this attribute is not applicable.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76530: usage context of syncLossTimeout

  Problem description:

  What is the usage context of syncLossTimeout? For example, i have one globalTimeDomain and three sub-domains. One Ethernet, and two CAN. Shall the synclossTimeout be set on the time-domain, or each of the sub-domains, or all three. This is not clear by the specification.

  suggestion:
  only allow it for the sub-domains

Agreed solution:

1) Make GlobalTimeDomain.syncLossTimeout optional in the model.
2) Describe that the syncLossTimeout shall be specified for slaves and introduce the following spec item:
TPS_SYST_xxxx: Applicability of syncLossTimeout
GlobalTimeDomain.syncLossTimeout shall be specified for GlobalTimeDomains that have an aggregated slave and for all other cases this attribute is not applicable.
–Last change on issue 76530 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.32 Specification Item TPS_SYST_02164

Trace References:

RS_SYST_00049

Content:

Only if the ISignalPort which the ISignalTriggering is referring to has no ISignalPort.ISignal Port.firstTimeout defined and the LdCom module is present, this ISignal shall be handled by LdCom.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76604: Update spec item to handle firstTimeout in LdCom

  Problem description:

  ———————————————
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  ———————————————
  Description/Motivation:

  In RfC# 74379 the ISignalPort.firstTimeout has been introduced. Unfortunately the introduction of a spec item similar to "[TPS_SYST_02027] LdCom: No ISignalPort.timeout reception timeout defined" has been forgotten.

  Agreed solution:

SysT: introduce spec item after TPS_SYST_02027

[TPS_SYST_xxxx1] LdCom: No ISignalPort.firstTimeout reception timeout defined
Only if the ISignalPort which the ISignalTriggering is referring to has no ISignalPort.firstTimeout defined and the LdCom module is present, this ISignal shall be handled by LdCom. (RS_SYST_00049)

BW-C-Level:

Application	Specification	Bus
1	4	1

1.33 Specification Item TPS_SYST_02165

Trace References:

RS_SYST_00042

Content:

The CanCommunicationConnector.pncWakeupDataMask should not be computed from the PncMapping.pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing CanCommunicationConnector.CanCommunicationConnector.pncWakeupDataMask will be bitwise ORed to obtain the value of CanNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.System.pncVectorOffset.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76718: Move notes from CanCommunicationConnector.pncWakeupDataMask and clarify

  Problem description:

  Name: Robert Sakretz
  Phone:
  Role: WP-M

  Description/Motivation:

The note of CanCommunicationConnector.pncWakeupDataMask is big and contains vague statements.
Pull the note to the document and enhance it.

a) The use-case for the statement "This attribute should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs." needs to be re-assessed:
I am not sure how such a use-case might work ...

b) the statement "calculated over the whole payload (8 Byte) of the NmPdu" is wrong. NmPdus can also be smaller than 8 byte!

Agreed solution:

- Add a specification item to chapter 3.3.1.4 after Table 3.21: CanCommunicationConnector and remove the description from the class table (CanCommunicationConnector.pncWakeupDataMask):
"TPS_SYST_xxx1: Derivation of CanNmPnFilterMaskByte
The pncWakeupDataMask should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing CanCommunicationConnector.pncWakeupDataMask will be bitwise ORed to obtain the value of CanNmPnFilterMaskByte.
Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.pncVectorOffset."

- Update upstream mapping of CanNmPnFilterMaskByte, CanNmPnFilterMaskByteIndex, and CanNmPnFilterMaskByteValue(remove "8 Byte"):
"For one EcuInstance all contributing CanCommunicationConnector.pncWakeupDataMask will be bitwise ORed to obtain aggregated pncWakeupDataMask value for this ECU. Since the pncWakeupDataMask is calculated over the whole payload of the NmPdu, the leading Bytes of this aggregated pncWakeupDataMask shall be ignored based on the System.pncVectorOffset value.
In order to get the CanNmPnFilterMaskByteIndex and CanNmPnFilterMaskByteValue for all the bytes aggregated pncWakeupDataMask shall be processed in a littleEndian way.
E.g. if pncVectorOffset = 2 and aggregated pncWakeupDataMask has the value 2^63 this will end up in a CanNmPnFilterMaskByte with CanNmPnFilterMaskByteIndex = 5 and CanNmPnFilterMaskByteValue = 128."

- Add a specification item to chapter 3.3.6.5 after Table 3.60: EthernetCommunicationConnector and remove the description from the class table (EthernetCommunicationConnector.pncFilterDataMask):
"TPS_SYST_xxx2: Derivation of UdpNmPnFilterMaskByte
The UdpNmPnFilterMaskByte should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing EthernetCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain the value of UdpNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.pncVectorOffset."

- Update upstream mapping of UdpNmPnFilterMaskByte, UdpNmPnFilterMaskByteIndex, and UdpNmPnFilterMaskByteValue (remove "8 Byte"):
"For one EcuInstance all contributing EthernetCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain aggregated pncFilterDataMask value for this ECU. Since the pncFilterDataMask is calculated over the whole payload of the NmPdu, the leading Bytes of this aggregated pncFilterDataMask shall be ignored based on the System.pncVectorOffset value.
In order to get the UdpNmPnFilterMaskByteIndex and UdpNmPnFilterMaskByteValue for all the bytes aggregated pncFilterDataMask shall be processed in a littleEndian way.
E.g. if pncVectorOffset = 2 and aggregated pncFilterDataMask has the value 2^63 this will end up in a UdpNmPnFilterMaskByte with UdpNmPnFilterMaskByteIndex = 5 and UdpNmPnFilterMaskByteValue = 128."

- Add a specification item to chapter 3.3.4.3 after Table 3.31: FlexrayCommunicationConnector and remove the description from the class table (FlexrayCommunicationConnector.pncFilterDataMask):
"TPS_SYST_xxx2: Derivation of FrNmPnFilterMaskByte
The FrNmPnFilterMaskByte should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing FlexrayCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain the value of FrNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of

System.pncVectorOffset."

- Update upstream mapping of FrNmPnFilterMaskByte, FrNmPnFilterMaskByteIndex, and FrNmPnFilterMaskByteValue (remove "8 Byte"):
"For one EcuInstance all contributing FlexrayCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain aggregated pncFilterDataMask value for this ECU. Since the pncFilterDataMask is calculated over the whole payload of the NmPdu, the leading Bytes of this aggregated pncFilterDataMask shall be ignored based on the System.pncVectorOffset value.
In order to get the FrNmPnFilterMaskByteIndex and FrNmPnFilterMaskByteValue for all the bytes aggregated pncFilterDataMask shall be processed in a littleEndian way.
E.g. if pncVectorOffset = 2 and aggregated pncFilterDataMask has the value 2^63 this will end up in a FrNmPnFilterMaskByte with FrNmPnFilterMaskByteIndex = 5 and FrNmPnFilterMaskByteValue = 128."
–Last change on issue 76718 comment 4–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.34 Specification Item TPS_SYST_02166

Trace References:

RS_SYST_00042

Content:

The UdpNmPnFilterMaskByte should not be computed from the PncMapping.pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing EthernetCommunicationConnector.EthernetCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain the value of UdpNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.System.pncVectorOffset.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76718: Move notes from CanCommunicationConnector.pncWakeupDataMask and clarify

Problem description:

Name: Robert Sakretz
Phone:
Role: WP-M

Description/Motivation:

The note of CanCommunicationConnector.pncWakeupDataMask is big and contains vague statements.
Pull the note to the document and enhance it.

a) The use-case for the statement "This attribute should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs." needs to be re-assessed:
I am not sure how such a use-case might work ...

b) the statement "calculated over the whole payload (8 Byte) of the NmPdu" is wrong. NmPdus can also be smaller than 8 byte!

Agreed solution:

- Add a specification item to chapter 3.3.1.4 after Table 3.21: CanCommunicationConnector and remove the description from the class table (CanCommunicationConnector.pncWakeupDataMask):
"TPS_SYST_xxx1: Derivation of CanNmPnFilterMaskByte
The pncWakeupDataMask should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing CanCommunicationConnector.pncWakeupDataMask will be bitwise ORed to obtain the value of CanNmPnFilterMaskByte.
Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.pncVectorOffset."

- Update upstream mapping of CanNmPnFilterMaskByte, CanNmPnFilterMaskByteIndex, and CanNmPnFilterMaskByteValue(remove "8 Byte"):
"For one EcuInstance all contributing CanCommunicationConnector.pncWakeupDataMask will be bitwise ORed to obtain aggregated pncWakeupDataMask value for this ECU. Since the pncWakeupDataMask is calculated over the whole payload of the NmPdu, the leading Bytes of this aggregated pncWakeup-

DataMask shall be ignored based on the System.pncVectorOffset value.
In order to get the CanNmPnFilterMaskByteIndex and CanNmPnFilterMaskByteValue for all the bytes aggregated pncWakeupDataMask shall be processed in a littleEndian way.
E.g. if pncVectorOffset = 2 and aggregated pncWakeupDataMask has the value 2^63 this will end up in a CanNmPnFilterMaskByte with CanNmPnFilterMaskByteIndex = 5 and CanNmPnFilterMaskByteValue = 128."

- Add a specification item to chapter 3.3.6.5 after Table 3.60: EthernetCommunicationConnector and remove the description from the class table (EthernetCommunicationConnector.pncFilterDataMask):
"TPS_SYST_xxx2: Derivation of UdpNmPnFilterMaskByte
The UdpNmPnFilterMaskByte should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing EthernetCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain the value of UdpNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.pncVectorOffset."

- Update upstream mapping of UdpNmPnFilterMaskByte, UdpNmPnFilterMaskByteIndex, and UdpNmPnFilterMaskByteValue (remove "8 Byte"):
"For one EcuInstance all contributing EthernetCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain aggregated pncFilterDataMask value for this ECU. Since the pncFilterDataMask is calculated over the whole payload of the NmPdu, the leading Bytes of this aggregated pncFilterDataMask shall be ignored based on the System.pncVectorOffset value.
In order to get the UdpNmPnFilterMaskByteIndex and UdpNmPnFilterMaskByteValue for all the bytes aggregated pncFilterDataMask shall be processed in a littleEndian way.
E.g. if pncVectorOffset = 2 and aggregated pncFilterDataMask has the value 2^63 this will end up in a UdpNmPnFilterMaskByte with UdpNmPnFilterMaskByteIndex = 5 and UdpNmPnFilterMaskByteValue = 128."

- Add a specification item to chapter 3.3.4.3 after Table 3.31: FlexrayCommunicationConnector and remove the description from the class table (FlexrayCommunicationConnector.pncFilterDataMask):
"TPS_SYST_xxx2: Derivation of FrNmPnFilterMaskByte

The FrNmPnFilterMaskByte should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing FlexrayCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain the value of FrNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.pncVectorOffset."

- Update upstream mapping of FrNmPnFilterMaskByte, FrNmPnFilterMaskByteIndex, and FrNmPnFilterMaskByteValue (remove "8 Byte"):
"For one EcuInstance all contributing FlexrayCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain aggregated pncFilterDataMask value for this ECU. Since the pncFilterDataMask is calculated over the whole payload of the NmPdu, the leading Bytes of this aggregated pncFilterDataMask shall be ignored based on the System.pncVectorOffset value.
In order to get the FrNmPnFilterMaskByteIndex and FrNmPnFilterMaskByteValue for all the bytes aggregated pncFilterDataMask shall be processed in a littleEndian way.
E.g. if pncVectorOffset = 2 and aggregated pncFilterDataMask has the value 2^63 this will end up in a FrNmPnFilterMaskByte with FrNmPnFilterMaskByteIndex = 5 and FrNmPnFilterMaskByteValue = 128."
–Last change on issue 76718 comment 4–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.35 Specification Item TPS_SYST_02167

Trace References:

RS_SYST_00042

Content:

The FrNmPnFilterMaskByte should not be computed from the PncMapping.pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing FlexrayCommunicationConnector.FlexrayCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain the value of FrNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload

of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.System.pncVectorOffset.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76718: Move notes from CanCommunicationConnector.pncWakeupDataMask and clarify

  Problem description:

  _____
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  _____
  Description/Motivation:

  The note of CanCommunicationConnector.pncWakeupDataMask is big and contains vague statements.
  Pull the note to the document and enhance it.

  a) The use-case for the statement "This attribute should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs." needs to be re-assessed:
  I am not sure how such a use-case might work ...

  b) the statement "calculated over the whole payload (8 Byte) of the NmPdu" is wrong. NmPdus can also be smaller than 8 byte!

  Agreed solution:

  - Add a specification item to chapter 3.3.1.4 after Table 3.21: CanCommunicationConnector and remove the description from the class table (CanCommunicationConnector.pncWakeupDataMask):
  "TPS_SYST_xxx1: Derivation of CanNmPnFilterMaskByte
  The pncWakeupDataMask should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs.

  Note that for one EcuInstance all contributing CanCommunicationConnector.pncWakeupDataMask will be bitwise ORed to obtain the value of CanNmPnFilterMaskByte.
  Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading

bytes which shall be ignored is equivalent to the value of System.pncVectorOffset."

- Update upstream mapping of CanNmPnFilterMaskByte, CanNmPnFilterMaskByteIndex, and CanNmPnFilterMaskByteValue(remove "8 Byte"):
"For one EcuInstance all contributing CanCommunicationConnector.pncWakeupDataMask will be bitwise ORed to obtain aggregated pncWakeupDataMask value for this ECU. Since the pncWakeupDataMask is calculated over the whole payload of the NmPdu, the leading Bytes of this aggregated pncWakeupDataMask shall be ignored based on the System.pncVectorOffset value.
In order to get the CanNmPnFilterMaskByteIndex and CanNmPnFilterMaskByteValue for all the bytes aggregated pncWakeupDataMask shall be processed in a littleEndian way.
E.g. if pncVectorOffset = 2 and aggregated pncWakeupDataMask has the value 2^63 this will end up in a CanNmPnFilterMaskByte with CanNmPnFilterMaskByteIndex = 5 and CanNmPnFilterMaskByteValue = 128."

- Add a specification item to chapter 3.3.6.5 after Table 3.60: EthernetCommunicationConnector and remove the description from the class table (EthernetCommunicationConnector.pncFilterDataMask):
"TPS_SYST_xxx2: Derivation of UdpNmPnFilterMaskByte
The UdpNmPnFilterMaskByte should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing EthernetCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain the value of UdpNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.pncVectorOffset."

- Update upstream mapping of UdpNmPnFilterMaskByte, UdpNmPnFilterMaskByteIndex, and UdpNmPnFilterMaskByteValue (remove "8 Byte"):
"For one EcuInstance all contributing EthernetCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain aggregated pncFilterDataMask value for this ECU. Since the pncFilterDataMask is calculated over the whole payload of the NmPdu, the leading Bytes of this aggregated pncFilterDataMask shall be ignored based on the System.pncVectorOffset value.
In order to get the UdpNmPnFilterMaskByteIndex and UdpNmPnFilterMaskByteValue for all the bytes aggregated pncFilterDataMask shall be processed in a littleEndian way.
E.g. if pncVectorOffset = 2 and aggregated pncFilterDataMask has the value 2^63 this will end up in a UdpNmPnFilterMaskByte with UdpNmPnFilterMaskByteIndex =

5 and UdpNmPnFilterMaskByteValue = 128."

- Add a specification item to chapter 3.3.4.3 after Table 3.31: FlexrayCommunicationConnector and remove the description from the class table (FlexrayCommunicationConnector.pncFilterDataMask):
"TPS_SYST_xxx2: Derivation of FrNmPnFilterMaskByte
The FrNmPnFilterMaskByte should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs.

Note that for one EcuInstance all contributing FlexrayCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain the value of FrNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.pncVectorOffset."

- Update upstream mapping of FrNmPnFilterMaskByte, FrNmPnFilterMaskByteIndex, and FrNmPnFilterMaskByteValue (remove "8 Byte"):
"For one EcuInstance all contributing FlexrayCommunicationConnector.pncFilterDataMask will be bitwise ORed to obtain aggregated pncFilterDataMask value for this ECU. Since the pncFilterDataMask is calculated over the whole payload of the NmPdu, the leading Bytes of this aggregated pncFilterDataMask shall be ignored based on the System.pncVectorOffset value.
In order to get the FrNmPnFilterMaskByteIndex and FrNmPnFilterMaskByteValue for all the bytes aggregated pncFilterDataMask shall be processed in a littleEndian way.
E.g. if pncVectorOffset = 2 and aggregated pncFilterDataMask has the value 2^63 this will end up in a FrNmPnFilterMaskByte with FrNmPnFilterMaskByteIndex = 5 and FrNmPnFilterMaskByteValue = 128."
–Last change on issue 76718 comment 4–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.36 Specification Item TPS_SYST_02168

Trace References:

none

Content:

The usage of CanFrameTriggering.CanFrameTriggering.txMask requires the support of COM Stack MetaData.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77574: [SysT] Necessity of Meta Data in case of dynamic CAN IDs

  Problem description:

  I think we need requirements similar to TPS_SYST_02151 to ensure that MetaData configuration is created (derived) for masked CAN IDs:

  TxMask: MDTs (MetaDataTypes) shall be added for variable SA/TA/Priority
  RxMask: MDTs (MetaDataTypes) shall be added for variable SA/TA

  Agreed solution:

  Add the following specification to chapter 6.7.3 Can specific description:

  [TPS_SYST_0xxx1] MetaData support required if CanFrameTriggering.txMask is used
  The usage of CanFrameTriggering.txMask requires the support of COM Stack MetaData.

  Please note that the MetaData support in [TPS_SYST_0xxx1] is required to calculate CAN-Ids at run-time.

  [TPS_SYST_0xxx2] MetaData support may be required if CanFrameTriggering.rxMask is used
  The usage of CanFrameTriggering.rxMask may require the support of COM Stack MetaData.

  Please note that the MetaData support in [TPS_SYST_0xxx2] is required if the upper layer is interested in the masked part of CAN-Id, e.g. J1939. In some cases the upper layer is not interested in the masked part of CAN-Id, e.g. for CanNm the MetaData is not required.
  –Last change on issue 77574 comment 6–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.37 Specification Item TPS_SYST_02169

Trace References:

none

Content:

The usage of CanFrameTriggering.CanFrameTriggering.rxMask may require the support of COM Stack MetaData.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77574: [SysT] Necessity of Meta Data in case of dynamic CAN IDs

  Problem description:

  I think we need requirements similar to TPS_SYST_02151 to ensure that MetaData configuration is created (derived) for masked CAN IDs:

  TxMask: MDTs (MetaDataTypes) shall be added for variable SA/TA/Priority
  RxMask: MDTs (MetaDataTypes) shall be added for variable SA/TA

  Agreed solution:

  Add the following specification to chapter 6.7.3 Can specific description:

  [TPS_SYST_0xxx1] MetaData support required if CanFrameTriggering.txMask is used
  The usage of CanFrameTriggering.txMask requires the support of COM Stack MetaData.

  Please note that the MetaData support in [TPS_SYST_0xxx1] is required to calculate CAN-Ids at run-time.

  [TPS_SYST_0xxx2] MetaData support may be required if CanFrameTriggering.rxMask is used
  The usage of CanFrameTriggering.rxMask may require the support of COM Stack MetaData.

  Please note that the MetaData support in [TPS_SYST_0xxx2] is required if the upper layer is interested in the masked part of CAN-Id, e.g. J1939. In some cases the upper layer is not interested in the masked part of CAN-Id, e.g. for CanNm the MetaData is not required.
  –Last change on issue 77574 comment 6–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.38 Specification Item TPS_SYST_02170

Trace References:

none

Content:

The Identifiable.category of the GeneralPurposeConnection is used to define the purpose of the relationship between the referenced PduTriggerings.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74995: [XCP] Clean up Chapters 8 and 10

  Problem description:

  XCP lacks the possibility to associate Rx-PDUs with Tx-PDUs of the same bus. Besides, a reference to a ComMChannel is missing in the EcuC that is used as an argument to Xcp_SetTransmissionMode() (which by the way is in the wrong section).

  Agreed solution:

  SWS XCP:
  Regarding the XCP configuration must be adapted such that there is a link between the Rx and Tx PDUs (XcpRxPdu/XcpTxPdu) that are used for request/response from one master.
  - Xcp_SetTransmissionMode() has a ComM channel as an argument, but the XCP does not know which PDUs belong to which ComM channel (or at least that is quite complicated to find out).

  The PS
  - Introduction of a new container (XcpCommunicationChannel) which references the corresponding Tx and Rx PDU (Tx PDU is mandatory, Rx PDU is optional) and the ComM channel they belong to.
  - Move Xcp_SetTransmissionMode() Spec Items SWS_Xcp_00844, SWS_Xcp_00848, SWS_Xcp_00849 and SWS_Xcp_00850 from section 8.4 to section 8.3.

  Create new CR for ECUC XML:
  Create new XCP configuration container:
  Container name: XcpCommunicationChannel

Container Parameter:
- XcpChannelTxPduRef: TX PDU (XcpTxPdu) reference (mandatory)
- XcpChannelRxPduRef: RX PDU (XcpRxPdu) reference (optional)
- XcpComMChannel: ComM channel (ComMChannel) reference (mandatory): which define what ComM channel which XCP belong to.

SysT:

- Introduce a new ARElement "GeneralPurposeConnection" that references to * PduTriggerings.
- Introduce a new category "XcpChannel" for "GeneralPurposeConnection" and define that in case of an "XcpChannel" the "GeneralPurposeConnection" is allowed to reference exactly two PduTriggerings (one TX and one RX Pdu).
- Introduce a constraint that defines that the PduTriggerings that are referenced by a "GeneralPurposeConnection" are defined on the same PhysicalChannel.
- Introduce a constraint that defines that the PduTriggerings that are referenced by "GeneralPurposeConnection" of category "XcpChannel" shall refer to Pdus of type "GeneralPurposeIPdu" with category "XCP"

In addition the following changes shall be done in the specification:
- Remove footnote on the first page of Section 6.3.
- Rework the first paragraph of Section 6.3 to the following: "The PDU Router is responsible only for the routing of IPdus (i.e., other types of Pdus are not routed by the PDU Router)."
- Remove footnote in Section 6.8.2.
–Last change on issue 74995 comment 25–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.39 Specification Item TPS_SYST_02171

Trace References:

RS_SYST_00054

Content:

The area within the payload Pdu that is secured is specified by the SecureCommunication Props.securedAreaOffset and SecureCommunicationProps.securedAreaLength. In case that these two attributes are not configured the complete payload Pdu is secured.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77090: [SecOC] Configuration of secured area within a Pdu

  Problem description:

  Currently it is not possible to configure a range/area within one Pdu which shall be secured. This possibility shall be added

  Agreed solution:

  SWS_SecOC:
  See attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4500
  ~change range from 1..2^32-1 to 0..2^32-1

  TPS_SystemTemplate:
  add the following optional attributes to SecureCommunicationProps:
  - securedAreaOffset (PositiveInteger) - This attribute defines the start position (offset in byte) of the area within the payload Pdu which will be secured
  - securedAreaLength (PositiveInteger) - This attribute defines the length in bytes of the area within the payload Pdu which will be secured

  Add the following specification item into chapter 6.3.2 SecuredIPdu

  [TPS_SYST_0XXX1] Secured Area in payload Pdu
  The area within the payload Pdu that is secured is specified by the securedAreaOffset and securedAreaLength. In case that these two attributes are not configured the complete
  payload Pdu is secured.

  [constr_xxx1] Existence of securedAreaOffset and securedAreaLength
  If the securedAreaOffset is defined then the securedAreaLength shall be defined as well and vice versa.
  –Last change on issue 77090 comment 29–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.40 Specification Item TPS_SYST_02172

Trace References:

RS_SYST_00054

Content:

If the SecuredIPdu.useAsCryptographicIPdu is set to false only the SecuredIPdu shall be

- mapped into a Frame by the PduToFrameMapping or
- assigned to SocketConnectionBundle or SocketConnection or
- assigned to ContainerIPdu

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77803: Enhance description of SecuredIPdu by big picture and example

  Problem description:

  The description of the SecuredIPdu is currently a bit sparse, and that leads to discussions how models shall be created and interpreted.

  It would therefore nice if we could clarify the modeling of SecuredIPdu as well as the authenticated IPdu.

  Is is necessary to assign Pdu ports to each of them?

  What about PduToFrameMapping, is it required for both Pdus? Rationale? How does the PduToFrameMapping typically look like? Why couldn't the existence of PduToFrameMapping be taken to deliver the information that is now formalized as useAsCryptographicPdu?

  The big picture such that both secured and authenticated IPdu need to be described in the system description independent of whether it is foreseen to actually put one or two Pdus on the bus should be described in more detail.

  An example would also be a welcome source of insight.

  I talked to Marek about this issue and he generally agrees. We will need to discuss this in the SystT group and prepare a proposed solution.

  Agreed solution:

  Add the following text after TPS_SYST_02149:

  The attribute useAsCryptographicIPdu decides whether one single Pdu or two Pdus are transferred on the communication bus. In either case always two IPdus shall
  be modeled:
  - SecuredIPdu with a PduTriggering

- payload IPdu with a PduTriggering

TPS_SysT_xxxx1 Modeling of SecuredIPdu in case useAsCryptographicIPdu is set to false
If the useAsCryptographicIPdu is set to false only the SecuredIPdu shall be
- mapped into a Frame by the PduToFrameMapping or
- assigned to SocketConnectionBundle or SocketConnection or
- assigned to ContainerPdu

TPS_SysT_xxxx2 Modeling of SecuredIPdu in case useAsCryptographicIPdu is set to true
If the useAsCryptographicIPdu is set to true then the SecuredIPdu and the payload IPdu shall be
- mapped into Frame(s) by the PduToFrameMapping or
- assigned to SocketConnectionBundle(s) or SocketConnection(s) or
- assigned to ContainerPdu(s)

Please note that [TPS_SYST_02059] defines that the PduTriggerings of the SecuredIPdu and PduTriggerings of the payload IPdu shall both reference IPdu-Ports.

In addition add object diagrams as examples that model the ARXML example that is available in Jira AP-2437.
–Last change on issue 77803 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.41 Specification Item TPS_SYST_02173

Trace References:

RS_SYST_00054

Content:

If the SecuredIPdu.useAsCryptographicIPdu is set to true then the SecuredIPdu and the SecuredIPdu.payloadIPdu shall be

- mapped into a Frame by the PduToFrameMapping or
- assigned to SocketConnectionBundle or SocketConnection or

- assigned to ContainerIPdu

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77803: Enhance description of SecuredIPdu by big picture and example

  Problem description:

  The description of the SecuredIPdu is currently a bit sparse, and that leads to discussions how models shall be created and interpreted.

  It would therefore nice if we could clarify the modeling of SecuredIPdu as well as the authenticated IPdu.

  Is is necessary to assign Pdu ports to each of them?

  What about PduToFrameMapping, is it required for both Pdus? Rationale? How does the PduToFrameMapping typically look like? Why couldn't the existence of PduToFrameMapping be taken to deliver the information that is now formalized as useAsCryptographicPdu?

  The big picture such that both secured and authenticated IPdu need to be described in the system description independent of whether it is foreseen to actually put one or two Pdus on the bus should be described in more detail.

  An example would also be a welcome source of insight.

  I talked to Marek about this issue and he generally agrees. We will need to discuss this in the SystT group and prepare a proposed solution.

  Agreed solution:

  Add the following text after TPS_SYST_02149:

  The attribute useAsCryptographicIPdu decides whether one single Pdu or two Pdus are transferred on the communication bus. In either case always two IPdus shall
  be modeled:
  - SecuredIPdu with a PduTriggering
  - payload IPdu with a PduTriggering

  TPS_SysT_xxxx1 Modeling of SecuredIPdu in case useAsCryptographicIPdu is set to false
  If the useAsCryptographicIPdu is set to false only the SecuredIPdu shall be
  - mapped into a Frame by the PduToFrameMapping or

- assigned to SocketConnectionBundle or SocketConnection or
- assigned to ContainerPdu

TPS_SysT_xxxx2 Modeling of SecuredIPdu in case useAsCryptographicIPdu is set to true
If the useAsCryptographicIPdu is set to true then the SecuredIPdu and the payload IPdu shall be
- mapped into Frame(s) by the PduToFrameMapping or
- assigned to SocketConnectionBundle(s) or SocketConnection(s) or
- assigned to ContainerPdu(s)

Please note that [TPS_SYST_02059] defines that the PduTriggerings of the SecuredIPdu and PduTriggerings of the payload IPdu shall both reference IPdu-Ports.

In addition add object diagrams as examples that model the ARXML example that is available in Jira AP-2437.
–Last change on issue 77803 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.42 Specification Item TPS_SYST_02174

Trace References:

none

Content:

The Initial Wait Phase for a ProvidedServiceInstance is configured with the SdServerConfig.initialOfferBehavior and the two attributes InitialSdDelayConfig.initialDelayMinValue and InitialSdDelayConfig.initialDelayMaxValue.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  ———————————

  Name: Robert Sakretz

Phone:
Role: WP-M

Description/Motivation:

The classes SdServerConfig and SdClientConfig are aggregated two times:
ProvidedServiceInstance.sdServerConfig
EventHandler.sdServerConfig
ConsumedServiceInstance.sdClientConfig
ConsumedEventGroup.sdClientConfig

This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
EventHandler.sdServerConfig.serverServiceMajorVersion

In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.

If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of

SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.43 Specification Item TPS_SYST_02175

Trace References:

none

Content:

The Repetition Wait Phase for a ProvidedServiceInstance is configured with the SdServer Config.initialOfferBehavior and the two attributes InitialSdDelayConfig.initialRepetitions Max and InitialSdDelayConfig.initialRepetitionsBaseDelay.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  _____

  Name: Robert Sakretz
  Phone:
  Role: WP-M
  _____

  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
  ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
  EventHandler.sdServerConfig.serverServiceMajorVersion

  In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

  My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
  So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

  Agreed solution:

  Add the following tables into the specification as explanation (see also attachment):
  https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

  Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.

Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.44 Specification Item TPS_SYST_02176

Trace References:

none

Content:

The Main Phase for a ProvidedServiceInstance is configured with the SdServerConfig.offerCyclicDelay attribute of SdServerConfig.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  ———————————————
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  ———————————————
  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
  ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
  EventHandler.sdServerConfig.serverServiceMajorVersion

  In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceIn-

stance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.45 Specification Item TPS_SYST_02177

Trace References:

none

Content:

The lifetime of a ProvidedServiceInstance is configurable with the SdServerConfig.ttl attribute of SdServerConfig.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  ———————————————
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  ———————————————
  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig

ConsumedServiceInstance.sdClientConfig
ConsumedEventGroup.sdClientConfig

This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
EventHandler.sdServerConfig.serverServiceMajorVersion

In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with

an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the

capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServer-Config. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received Sub-scribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClient-Config. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.46 Specification Item TPS_SYST_02178

Trace References:

none

Content:

The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.SdServerConfig.requestResponseDelay.

The actual delay will be randomly chosen between the RequestResponseDelay.maxValue and RequestResponseDelay.minValue.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  Name: Robert Sakretz
  Phone:
  Role: WP-M

  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
  ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
  EventHandler.sdServerConfig.serverServiceMajorVersion

  In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

  My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
  So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

  Agreed solution:

  Add the following tables into the specification as explanation (see also attachment):
  https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

  Add the following explanations as specification items in addition to the new tables into the specification:

  On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
  - Initial Wait Phase:
  The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue.

When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService

entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.47 Specification Item TPS_SYST_02179

Trace References:

none

Content:

A Capability Record (key/value pair) on the Server side is configurable with the SdServer Config.capabilityRecord and the two attributes TagWithOptionalValue.key and TagWithOptionalValue.value.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  Name: Robert Sakretz
  Phone:
  Role: WP-M

  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
  ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
  EventHandler.sdServerConfig.serverServiceMajorVersion

  In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

  My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
  So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

  Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery

communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.

- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.48 Specification Item TPS_SYST_02180

Trace References:

none

Content:

The switching between IP-Unicast and IP-Multicast is guided by the server with the EventHandler.EventHandler.multicastThreshold attribute and by the number of subscribed clients to the EventHandler.

The Server will change the transmission of events to Multicast if the EventHandler.multicastThreshold of the corresponding EventHandler is reached by the number of subscribed clients. If the number of subscribed clients is smaller then the configured EventHandler.multicastThreshold, the transmission of events takes place via unicast communication.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  ———————————————
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  ———————————————
  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig

EventHandler.sdServerConfig
ConsumedServiceInstance.sdClientConfig
ConsumedEventGroup.sdClientConfig

This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
EventHandler.sdServerConfig.serverServiceMajorVersion

In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDe-

lay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.

A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.49 Specification Item TPS_SYST_02181

Trace References:

none

Content:

The lifetime of an event subscription acknowledge message is configurable with the Sd ServerConfig.ttl attribute of SdServerConfig that is aggregated by an EventHandler in the role EventHandler.sdServerConfig.

If the time that is configured by SdServerConfig.ttl expires the event subscription acknowledge is canceled.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  _____
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  _____
  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
  ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
  EventHandler.sdServerConfig.serverServiceMajorVersion

  In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

  My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
  So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

  Agreed solution:

  Add the following tables into the specification as explanation (see also attachment):
  https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

  Add the following explanations as specification items in addition to the new tables into the specification:

  On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
  - Initial Wait Phase:

The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.

If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.50 Specification Item TPS_SYST_02182

Trace References:

none

Content:

The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEvent Group message that was triggered by a multicast ServiceOffer by the configured SdServer Config.SdServerConfig.requestResponseDelay that is aggregated by the EventHandler in the role EventHandler.sdServerConfig.

The actual delay will be randomly chosen between the RequestResponseDelay.maxValue and RequestResponseDelay.minValue.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  _____
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  _____
  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
  ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
  EventHandler.sdServerConfig.serverServiceMajorVersion

  In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

  My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.

So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment): https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name

of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be

configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClient-Config. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.51 Specification Item TPS_SYST_02183

Trace References:

none

Content:

The Initial Wait Phase for a ConsumedServiceInstance is configured with the SdClient Config.initialFindBehavior and the two attributes InitialSdDelayConfig.initialDelayMinValue and InitialSdDelayConfig.initialDelayMaxValue.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  ———————————————
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  ———————————————
  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig

ConsumedEventGroup.sdClientConfig

This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
EventHandler.sdServerConfig.serverServiceMajorVersion

In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.

- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.52 Specification Item TPS_SYST_02184

Trace References:

none

Content:

The Repetition Wait Phase for a ConsumedServiceInstance is configured with the SdClientConfig.initialFindBehavior and the two attributes InitialSdDelayConfig.initialRepetitionsMax and InitialSdDelayConfig.initialRepetitionsBaseDelay.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

Problem description:

Name: Robert Sakretz
Phone:
Role: WP-M

Description/Motivation:

The classes SdServerConfig and SdClientConfig are aggregated two times:
ProvidedServiceInstance.sdServerConfig
EventHandler.sdServerConfig
ConsumedServiceInstance.sdClientConfig
ConsumedEventGroup.sdClientConfig

This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
EventHandler.sdServerConfig.serverServiceMajorVersion

In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.

- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.

Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.53 Specification Item TPS_SYST_02185

Trace References:

none

Content:

The lifetime of a ConsumedServiceInstance is configurable with the SdClientConfig.ttl attribute of SdClientConfig.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  ———————————————
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  ———————————————
  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
  ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
  EventHandler.sdServerConfig.serverServiceMajorVersion

  In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

  My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
  So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

  Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery

communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.

- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.54 Specification Item TPS_SYST_02186

Trace References:

none

Content:

A Capability Record (key/value pair) on the Client side is configurable with the SdClient Config.capabilityRecord and the two attributes TagWithOptionalValue.key and TagWithOptionalValue.value.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  ———————————
  Name: Robert Sakretz
  Phone:
  Role: WP-M
  ———————————
  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:

ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
EventHandler.sdServerConfig.serverServiceMajorVersion

In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.

- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServer-

Config. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.55 Specification Item TPS_SYST_02187

Trace References:

none

Content:

The lifetime of an event subscription is configurable with the SdClientConfig.ttl attribute of SdClientConfig that is aggregated by an ConsumedEventGroup in the role ConsumedEventGroup.sdClientConfig.

If the time that is configured by SdClientConfig.ttl expires the event subscription is canceled.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

Name: Robert Sakretz
Phone:
Role: WP-M

Description/Motivation:

The classes SdServerConfig and SdClientConfig are aggregated two times:
ProvidedServiceInstance.sdServerConfig
EventHandler.sdServerConfig
ConsumedServiceInstance.sdClientConfig
ConsumedEventGroup.sdClientConfig

This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
EventHandler.sdServerConfig.serverServiceMajorVersion

In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the

initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.

- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.56 Specification Item TPS_SYST_02188

Trace References:

none

Content:

The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.SdClientConfig.requestResponseDelay that is aggregated by the ConsumedEventGroup in the role ConsumedEventGroup.sdClientConfig.

The actual delay will be randomly chosen between the RequestResponseDelay.maxValue and RequestResponseDelay.minValue.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75251: Specify when attributes of SdServerConfig and SdClientConfig are applicable

  Problem description:

  Name: Robert Sakretz
  Phone:
  Role: WP-M

  Description/Motivation:

  The classes SdServerConfig and SdClientConfig are aggregated two times:
  ProvidedServiceInstance.sdServerConfig
  EventHandler.sdServerConfig
  ConsumedServiceInstance.sdClientConfig
  ConsumedEventGroup.sdClientConfig

  This leads to the situation that all of the enclosed attributes (e.g. serverServiceMajorVersion) are possible to be defined in both roles:
  ProvidedServiceInstance.sdServerConfig.serverServiceMajorVersion
  EventHandler.sdServerConfig.serverServiceMajorVersion

  In many cases it is "obvious" that only one path makes sense, but to my understanding this is not specified.

  My expectation would be to define a sudoku table listing all possible attribute paths and specifying which path is applicable for which parent.
  So which attributes are allowd at the ServiceInstance and which attributes are allowed at the EventHandler/ConsumedEventGroup.

  Agreed solution:

Add the following tables into the specification as explanation (see also attachment):
https://svn.autosar.org/repos/work/09_WorkPackages/WP-M/06_Subgroups/01_SystemTemplate/99_Miscellaneous/RfC# 75251.xlsx

Add the following explanations as specification items in addition to the new tables into the specification:

On the ProvidedServiceInstance the following SOME/IP Service Discovery communication behavior is configurable:
- Initial Wait Phase:
The Initial Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first OfferService entry will be sent out.
- Repetition Wait Phase:
The Repetition Wait Phase for a ProvidedServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered. If intialRepetitionMax is configured to 0 the Repetition Phase will be skipped and the Main Phase will be entered.
- Main Phase:
The Main Phase for a ProvidedServiceInstance is configured with the offerCyclicDelay attribute of SdServerConfig. The OfferService entry will be sent cyclically with an interval that is defined with offerCyclicDelay.
- TTL for Offer Service Entries:
The lifetime of a ProvidedServiceInstance is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the ProvidedServiceInstance is no longer offered.
- Request Response Delay:
The Server will delay the OfferService answer to a received multicast FindService entry by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.
- Server Capability Records:
SOME/IP allows to specify additional information about the ProvidedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration. A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value.

On the ConsumedServiceInstance the following SOME/IP Service Discovery

communication behavior is configurable:
- Initial Wait Phase
The Initial Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue. When a calculated random timer based on these min and max values expires, the first FindService entry will be sent out.
Repetition Phase
The Repetition Wait Phase for a ConsumedServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay.
If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry. If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered.
Main Phase
In the Main Phase no further FindService entries are send by the client.
- TTL for Find Service Entries
The lifetime of a ConsumedServiceInstance is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the FindService entry shall be considered not existing.
- Client Capability Records:
SOME/IP allows to specify additional information about the ConsumedServiceInstance with the Configuration Option that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.
A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value.

On the ProvidedServiceInstance.eventHandler the following can be configured:
- TTL for SubscribeEventGroupAck Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdServerConfig. If the time that is configured by ttl expires the event subscription is canceled.
- RequestResponseDelay:
The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SdServerConfig.requestResponseDelay. The actual delay will be randomly chosen between the maxValue and minValue.

On the ConsumedServiceInstance.consumedEventGroup the following can be configured:
- TTL for SubscribeEventGroup Entries:
The lifetime of a event subscription is configurable with the ttl attribute of SdClientConfig. If the time that is configured by ttl expires the event subscription is canceled.

- RequestResponseDelay
The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SdClientConfig.requestResponseDelay.
The actual delay will be randomly chosen between the maxValue and minValue.
–Last change on issue 75251 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.57 Specification Item TPS_SYST_02189

Trace References:

RS_SYST_00054

Content:

The SecuredIPdu.useSecuredPduHeader shall be set to a value other than SecuredPdu HeaderEnum.noHeader if the length of the payload Pdu is dynamic and is transmitted over a network which may insert padding bytes depending on the length (e.g. CANFD, Flexray).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77336: [SECOC] Dynamic length PDUs (Container) not possible / clear

  Problem description:

  The SecOC SWS does not make any assumptions or restrictions about dynamic length PDUs. But with parameter SecOCAuthDataFreshnessStartPosition it could be impossible to really use dynamic length in SecOC.

  Additionally in SystemTemplate TPS the constraint constr_3139 talks only about some restrictions when dynamic length IPDUs are used. This would imply dynamic length IPDUs should be possible.

  Without dynamic length support it is also not possible to support securing complete IDPUM Containers.
  One possible solution could be (if dynamic length should not be supported by SecOC) to add a configuration option per IPDUM Container to send always maximum length (padding with 0) to have the static length again.
  If dynamic length shall be supported by SecOC a further problem will be the CAN-FD padding.

Agreed solution:

==========
AUTOSAR 4.3.1
==========

_____-
SRS SecOC
_____-

* Add new section next to 6.1.3.5 and new requirement [SRS_SecOC_xxxx1] Support of padding at lower layer modules and dynamic length Authentic I-PDUs.
* Description: The SecOC module shall be applicable for the use cases with padding at lower layer modules and with dynamic length Authentic I-PDUs.
* Rationale: At receiver side, received Secured I-PDU containing dynamic length Authentic I-PDU may also conatin padding bytes (added by lower layer modules of sender side, to fit to bus-specific L-PDU length constraints, e.g. CAN FD and FlexRay). In such case, receivers cannot identify number of bytes / byte position of the received payload.
* Use Case: dynamic length PDU on CAN FD and FlexRay
* Dependencies: [SRS_SecOC_00012]
* Supporting Material: -

Note: According to CM, RS Main should be referred. But upstream requirements in current (R4.3.0) SRS SecOC are from RS Features, and appropriate requirements are not available in RS Main.
(If we use RS Features, at least [RS_BRF_01568] [RS_BRF_01649] [RS_BRF_01712] [RS_BRF_01716] [RS_BRF_01752] [RS_BRF_02035] [RS_BRF_02036] [RS_BRF_02037] could be mapped to this requirements)

_____-
SWS SecOC
_____-

a1) Adapt Figure 4 in sec. 7.1.1.1

Change from
< (Figure of "Secured I-PDU = Authentic I-PDU | Freshness Value | Authenticator")
< Figure 4: Secured I-PDU contents
to
> (Figure of "Secured I-PDU = Secured I-PDU Header (optional) | Authentic I-PDU |

Freshness Value (optional) | Authenticator")
> Figure 4: Secured I-PDU contents
==> to be done as RfC # 77807, not handled in this RfC.

a2) Add new requirements regarding the Secured I-PDU Header and related behavior

Add new requirement [SWS_SecOC_xxxx2] to define the Secured I-PDU Header
> The Secured I-PDU Header shall indicate the length of the Authentic I-PDU in bytes. The length of the Header shall be configurable by the parameter SecOCAuthPduHeaderLength.
> ()
> Note: the SecOC supports combined usage of authentication data in a separate message (secured PDU collection) and Secured I-PDU Header. Also the SecOC covers dynamic length Authentic I-PDU.

Add new requirement [SWS_SecOC_xxxx3] for behavior at transmission (construction) of Secured I-PDUs
> For a Tx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall add the Secured I-PDU Header to the Secured I-PDU with the length of the Authentic I-PDU within the Secured I-PDU, to handle dynamic Authentic I-PDU.
> ()
> Note: Primary purpose of this Header is to indicate the position of Freshness Value and Authenticator in Secured I-PDUs with dynamic length Authentic I-PDU.
> Also some buses which cannot select arbitrary length of L-PDU (e.g. CAN FD and FlexRay) require this Header, because the position of Freshness Value and Authenticator is not always at the end of the Secured I-PDU, as lower layer modules (e.g. CanIf and FrIf) may add bus-specific padding bytes after processing at SecOC (then the L-PDU containing the Secured I-PDU with padding will be: Secured I-PDU = Secured I-PDU Header | Authentic I-PDU | Freshness Value | Authenticator | Bus-specific padding).

Add new requirement [SWS_SecOC_xxxx4] for behavior at reception of Secured I-PDUs
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall assume Secured I-PDU Header shall be available in the Secured I-PDU, to handle dynamic Authentic I-PDU.
> ()

Add new requirement [SWS_SecOC_xxxx5] for behavior at reception of Secured I-PDUs, the Header tells it's longer than the maximum length of the PDU
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0 and the length of Authentic I-PDU in the Header is longer than configured length (in case of dynamic length IPdus (containing a dynamical length signal), this value indicates the maximum data length) of the Authentic I-PDU, the SecOC module shall discard the I-PDU. In such case with SecOC_StartOfReception, BUFREQ_E_NOT_OK shall be returned (see [SWS_COMTYPE_00012]).
> ()
> Note: SecOC_RxIndication has no return value.

Add new requirement [SWS_SecOC_xxxx6] for behavior at reception of Secured I-PDUs, the Header tells it's shorter than received I-PDU length (ignoring the padding at the end)
> For a Rx Secured I-PDU with SecOCAuthPduHeaderLength > 0, the SecOC module shall process Secured I-PDU Header, Authentic I-PDU (with the length specified by the Header), Freshness Value and Authenticator of the Rx Secured I-PDU. The rest of bytes in the Secured I-PDU shall be discarded.
> ()

a3) Add a configuration parameter to SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPdu, SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection/SecOCRxAuthenticPdu, SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPdu and SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPduCollection/SecOCTxAuthenticPdu to enable/disable Secured I-PDU Header per I-PDU
* SWS Item: ECUC_SecOC_xxxx1
* Name: SecOCAuthPduHeaderLength
* Description:
* This parameter indicates the length (in bytes) of the Secured I-PDU Header in the Secured I-PDU. The length of zero means there's no header in the PDU.
* Multiplicity: 0..1
* Range: 0..4
* Default: 0

a4) (removed)

a8) Update layout definition (construction) for Secured I-PDUs

Change [SWS_SecOC_00037]
from
< [SWS_SecOC_00037]
< The SecOC module shall construct the Secured I-PDU by adding the Freshness Value and the Authenticator to the Authentic I-PDU.
< (SRS_SecOC_00006)
< Note: The Freshness Counter and the Authenticator included as part of the Secured I-PDU may be truncated per configuration specific to the identifier of the Secured I-PDU. The scheme for the Secured I-PDU looks as follows:
< SecuredPDU = AuthenticIPDU | FreshnessValue [SecOCFreshnessValueTxLength] | Authenticator [SecOCAuthInfoTxLength]
to
> [SWS_SecOC_00037]
> The SecOC module shall construct the Secured I-PDU by adding the Secured I-PDU Header (optional), the Freshness Value (optional) and the Authenticator to the Authentic I-PDU.
> The scheme for the Secured I-PDU (includes the order in which the contents are structured in the Secured I-PDU) shall be compliant with below:
> SecuredPDU = SecuredIPDUHeader (optional) | AuthenticIPDU | FreshnessValue [SecOCFreshnessValueTxLength] (optional) | Authenticator [SecOCAuthInfoTxLength]
> (SRS_SecOC_00006)
> Note: The Freshness Counter and the Authenticator included as part of the Secured I-PDU may be truncated per configuration specific to the identifier of the Secured I-PDU. Also Freshness Value may be a part of Authentic I-PDU (see [SWS_SecOC_00219]).
==> to be done as RfC # 77807, not handled in this RfC.

a9) Add new constraints and notes after [SWS_SecOC_00219]:
> [constr_xxxx1] All signals before SecOCAuthDataFreshnessStartPosition within the Secured I-PDU shall have static length.
> Note: SecOC can use a part of the Authentic I-PDU as freshness when SecOCUseAuthDataFreshness=true, only if the part of the Authentic I-PDU to be used as the freshness is always available at same position in the Authentic I-PDU.

> [constr_xxxx2] Any container I-PDU which contains multiple contained I-PDUs shall be set SecOCUseAuthDataFreshness=false.
> Note: For container PDUs, normally it cannot be ensured which PDU will be put in which position (depends on various timing and trigger conditions). Therefore, con-

tainer I-PDUs with multiple contained I-PDUs cannot have FV within the Authentic I-PDU.

a10) Adapt Figure 5
==> to be done as RfC # 77807, not handled in this RfC.

a11) Adapt SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection/SecOCRxAuthenticPdu and SecOC/SecOCTxPduProcessing/SecOCTxSecuredPduLayer/SecOCTxSecuredPduCollection/SecOCTxAuthenticPdu

Change the description of SecOCRxAuthenticPdu (ECUC_SecOC_00061)
from
< This container specifies the Authetic Pdu that is received by the SecOC module from the PduR.
to
> This container specifies the PDU (that is received by the SecOC module from the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.

Change the description of SecOCTxAuthenticPdu ECUC_SecOC_00072
from
< This container specifies the Authetic Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated.
to
> This container specifies the PDU (that is transmitted by the SecOC module to the PduR) which contains the Secured I-PDU Header and the Authentic I-PDU.

———————-
TPS System Template (SysT)
———————-

a5) Add new attribute to SecuredIPdu (Table 6.46) which enables SecOCAuthPduHeaderLength>0
* Attribute: useSecuredPduHeader
* Type: SecuredPduHeaderEnum
* Mul.: 0..1
* Kind: attr
* Desc: This attribute defines the size of the header which is inserted into the SecuredIPdu. If this attribute is set to anything but noHeader, the SecuredIPdu contains the Secured I-PDU Header to indicate the length of the AuthenticIPdu. The

AuthenticIPdu contains the original payload, i.e. the secured data.

SecuredPduHeaderEnum
- noHeader
- securedPduHeader08Bit
- securedPduHeader16Bit
- securedPduHeader32Bit
Desc: Defines the header which will be inserted into the SecuredIPdu.

a6) Change the description of IPduPort.rxSecurityVerification in Table 6.3: IPduPort:
This attribute defines the bypassing of signature authentication or MAC verification in the receiving ECU.
If not defined or set to true the signature authentication or MAC verification shall be performed for the SecuredIPdu.
If set to false the signature authentication or MAC verification shall not be performed for the SecuredIPdu.

Removed [constr_3139].

TPS_SysT_xxxx2: Setting of useSecuredPduHeader attribute
The useSecuredPduHeader shall be set to a value other than noHeader if the length of the payload Pdu is dynamic and is transmitted over a network which may insert padding bytes depending on the length (e.g. CANFD, Flexray).

Add a note below TPS_SysT_xxxx2:

Please note that the dynamic-length Pdu can be an ISignalIPdu that contains a SystemSignal with dynamicLength set to true. In general it is not possible to run diagnostics on fixed-length Pdus. Therefore, there is a probability that at least a subset of DcmIPdus and UserDefinedIPdus can have dynamic length.

a7) Add upstream mapping between useSecuredPduHeader (SysT) and SecOCAuthPduHeaderLength (EcuC) in C.1.4 SecOc Mapping

b4) Add upstream mapping between rxSecurityVerification (SysT) and SecOCSecuredRxPduVerification (EcuC) in C.1.4 SecOc Mapping
Mapping rule: SecOCSecuredRxPduVerification is True if rxSecurityVerification is not defined, otherwise SecOCSecuredRxPduVerification = rxSecurityVerification

———————-
SRS SecOC
———————-

* Add new section next to 6.1.3.5 (or 6.2.1.1) and new requirement [SRS_SecOC_xxxx2] Support of capability to extract Authentic I-PDU without Authentication
* Description: The SecOC module shall be capable to extract Authentic I-PDU from Secured I-PDU, without Authentication.
* Rationale: SecOC can be used as an extractor of Authentic I-PDU from Secured I-PDU, to enable low latency GW behavior when a part of downstream communication clusters doesn't require authentication of PDUs.
* Use Case: Gateway
* Dependencies: [SRS_SecOC_00025]
* Supporting Material: -

Note: According to CM, RS Main should be referred. But upstream requirments in current (R4.3.0) SRS SecOC are from RS Features, and appropriate requirements are not available in RS Main.
(If we use RS Features, at least [RS_BRF_02035] [RS_BRF_02036] [RS_BRF_02037] could be mapped to this requirements)

———————-
SWS SecOC
———————-

b1) Remove [constr_3139] (not [constr_3193] – sorry, constr_3193 is typo in my comment # 10)

b2) Add new requirements regarding skipped authentication behavior at SecOC (just remove FV/MAC from Secured I-PDU)

* Add new section "Extracting Authentic I-PDU without Authentication at SecOC" or "Skipping Authentication for Secured I-PDUs at SecOC"
* Add new requirement [SWS_SecOC_xxxx7] for behavior of SecOC at reception of Secured I-PDUs without Authentication
> For a Rx Secured I-PDU with SecOCSecuredRxPduVerification=false, the SecOC module shall extract the Authentic I-PDU using the length specified by the Secured I-PDU Header without Authentication.
> ()

b3) Add a configuration parameter to SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPdu and SecOC/SecOCRxPduProcessing/SecOCRxSecuredPduLayer/SecOCRxSecuredPduCollection to control

authentication behavior at SecOC

* SWS Item: ECUC_SecOC_xxxx3
* Name: SecOCSecuredRxPduVerification
* Description: This parameter defines whether the signature authentication or MAC verification shall be performed on this Secured I-PDU. If set to false, the SecOC module extracts the Authentic I-PDU from the Secured I-PDU without verification.
* Multiplicity: 1
* Type: EcucBooleanParamDef
* Default value: false
* Post-Build Variant Value: true
* Value Configuration Class:
* Pre-compile time: X All Variants
* Scope / Dependency: scope: local
–Last change on issue 77336 comment 69–

BW-C-Level:

Application	Specification	Bus
1	4	4