Document Title	Specification of Demands and Constraints on Base Software
Document Category	Software specification (SWS)
Document UID	1120
Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Change of document type from TR (Technical Report) to SWS (SW Specification) • New contribution 7.1.1 Security Events • Contributions on MACsec, Trusted Platform, ARTI Tracing in Sec. 7 • new subsection on MACsec default values in Sec. 10 • Introduction of common text for non applicable (sub-) sections
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes
2024-11-27	R24-11	AUTOSAR Release Management	• Initial Release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 5
2 Acronyms and Abbreviations 6
3 Related documentation 7
4 Constraints and assumptions 8
4.1 Known limitations . . . 8
5 Dependencies to Functional Clusters 9
6 Requirements Tracing 10
7 Functional specification 11
7.1 Reporting . . . 11
7.1.1 Security Events . . . 11
7.1.1.1 Ethernet . . . 12
7.1.1.2 TCP/IP . . . 13
7.1.1.3 MACsec . . . 15
7.1.1.4 Trusted Platform . . . 17
7.1.2 Log Messages . . . 17
7.1.3 Violation Messages . . . 18
7.1.4 Production Errors . . . 18
7.2 Trusted Platform . . . 18
7.2.1 Handling of failed authenticity checks . . . 20
7.2.1.1 Monitor Mode . . . 21
7.2.1.2 Strict Mode . . . 21
7.3 MACsec . . . 22
7.4 Demands and constraints for ARTI tracing . . . 23
7.4.1 Tracing of tasks . . . 26
7.4.2 Tracing of processes . . . 30
8 API Specification 34
9 Service Interfaces 35
10 Configuration 36
10.1 Default Values . . . 36
10.1.1 Default values for TCP parameters . . . 36
10.1.2 Default values for UDP parameters . . . 37
10.1.3 Default values for ICMPv4 parameters . . . 38
10.1.4 Default values for ICMPv6 parameters . . . 38
10.1.5 Default values for ARP parameters . . . 38
10.1.6 Default values for AutoIP parameters . . . 39
10.1.7 Default values for IPv4 fragmentation parameters . . . 39
10.1.8 Default values for DHCPv6 parameters . . . 39

10.1.9 Default values for NDP parameters 40
10.1.10 Default values for MACsec parameters 42
10.2 Semantic Constraints 42
A Mentioned Manifest Elements 43
B Demands and constraints on Base Software (normative) 56
C Platform Extension Interfaces (normative) 57
D Not implemented requirements 58
E History of Constraints and Specification Items 59
E.1 Change History of this document according to AUTOSAR Release R26-11 59
E.1.1 Added Specification Items in R26-11 59
E.1.2 Changed Specification Items in R26-11 62
E.1.3 Deleted Specification Items in R26-11 62

1 Introduction and functional overview

This document describes common aspects of usage of Adaptive Platform functionality towards Base SW, such as the OS or other kinds of "lower layer" SW. For the latter this may include requirements, proposed default settings of configuration parameters, usage constraints and alike. The previously used term 'common' indicates that the usage aspects do not reflect a single functional cluster (FC) perspective but instead addresses Base SW considerations potentially used by several functional clusters. In contrast Base SW aspects related to individual FC usage are covered by the respective SWS documents.

This document's main purpose is to assist different roles for setting up an development AP project such as the integrator. As such the Base SW topics in this document are not necessarily connected on their technical subjects. The focus of the document shall not be mixed with [1, AUTOSAR OSI interface specification], the latter addresses the interface from Adaptive Application perspective.

In contrast to many other AUTOSAR software specifications, this document is not related to a specific SW component (such as AP functional cluster).

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the Specification of Base Software that are not included in the [2, AUTOSAR glossary].

Abbreviation / Acronym	Description
ACK	Acknowledgement - message type of TCP providing confirmation about receiving the request to initiate the graceful shutdown or setup of a TCP connection
Base SW	Base Software - lower layer software that provides services or notifications to the Adaptive Platform, such as the operating system, hardware drivers, virtualization software, ...
DAD	Duplicate Address Detection
DLT	Diagnostics Log and Trace - a Log and Trace daemon implementation of the [3, LT protocol]
IETF RFC	Internet Engineering Task Force - Request For Comment: is a standard-like specification of protocols related to TCP/IP protocol suite
FIN	Finish - message type of TCP to initiate the graceful shutdown of a TCP connection
FINWAIT-2	TCP connection state at the communication peer after receiving the acknowledgement message (ACK) for connection termination
MTU	Maximum Transmission Unit
NDP	Neighbor Discovery Protocol
SYN	Synchronization - message type of TCP to initiate the setup of a TCP connection

Table 2.1: Acronyms and abbreviations used in the scope of this document

3 Related documentation

[1] Specification of Operating System Interface
AUTOSAR_AP_SWS_OperatingSystemInterface

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Log and Trace Protocol Specification
AUTOSAR_FO_PRS_LogAndTraceProtocol

[4] Technical Report on Security Events Specification
AUTOSAR_FO_TR_SecurityEventsSpecification

[5] General Specification of Adaptive Platform
AUTOSAR_AP_SWS_General

[6] Explanation of Adaptive Platform Design
AUTOSAR_AP_EXP_PlatformDesign

[7] Explanation of MACsec and MKA Protocols implementation and configuration guidelines
AUTOSAR_AP_EXP_MACsec

[8] Requirements on MACsec
AUTOSAR_FO_RS_MACsec

[9] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

[10] Specification of TCP/IP Stack
AUTOSAR_CP_SWS_Tcplp

4 Constraints and assumptions

4.1 Known limitations

This specification does not specify a functional cluster. This chapter does not apply.

5 Dependencies to Functional Clusters

This specification does not specify a functional cluster. This chapter does not apply.

6 Requirements Tracing

There are no requirements yet for tracing.

7 Functional specification

7.1 Reporting

7.1.1 Security Events

AUTOSAR defines a set of security events for different technologies (see [4, FO TR SecurityEventsSpecification] for an overview). Some of these Security Events cannot be raised by the Adaptive AUTOSAR stack itself, because their functionality is implemented below the Adaptive Platform, on the level of the base software. This section lists all Security Events that can be raised by the Base Software.

[SWS_APBSW_00001] Security Events for Base Software

Status: DRAFT

⌈The Base Software shall implement the Security Events defined in [SWS_APBSW_00002] and raise them to the Adaptive Intrusion Detection System Manager (AIdsM).⌋

[SWS_APBSW_00002] Security events for ap_core

Status: DRAFT
Previous identifiers: SWS_EM_02588

⌈

Name	*Description*	*ID*
SEV_ARP_IP_ADDR_CONFLICT	Received local IP address in ARP reply for different MAC.	10
SEV_TCP_DROP_INV_DEST_PORT	Dropped TCP packet because of invalid destination TCP-Port.	11
SEV_UDP_DROP_INV_DEST_PORT	Dropped UDP packet because of invalid destination UDP-Port.	12
SEV_IPV4_DROP_INV_DEST_ADDR	Dropped datagram because of invalid destination IPV4 address.	13
SEV_IPV6_DROP_INV_DEST_ADDR	Dropped datagram because of invalid destination IPV6 address.	14
SEV_ETH_DROP_UNKNOWN_ETHERTYPE	An ethernet datagram was dropped due the Ethertype is not known.	15
SEV_ETH_DROP_VLAN_DOUBLE_TAG	An ethernet datagram was dropped due to double VLAN tag.	16
SEV_ETH_DROP_INV_VLAN	An ethernet datagram was dropped due to an invalid Crtl Idx/VLAN.	17
SEV_ETH_DROP_MAC_COLLISION	Ethernet datagram was dropped because local MAC was same as source MAC in an incoming frame.	18
SEV_MKA_AUTHENTICATION_FAILURE	Event triggered when the authentication during the MKA communication has failed (wrong CKN/CAK).	78
SEV_MKA_TIMEOUT	Event triggered when the timeout for the MKA communication has expired.	79
SEV_MKA_PORT_NOT_ENABLED	Event triggered when the indicated port for the MKA communication is not enable.	80
SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED	Event triggered when there is no Cipher Suite supported.	81
SEV_MKA_PORT_NUMBER_CHANGE	Event triggered when during the MKA communication the port number has changed.	82

▽

△

Name	Description	ID
SEV_EXEC_SW_COMPONENT_INTEGRITY_CHECK_FAILED	The integrity check of a SW component has failed.	99

⌋

7.1.1.1 Ethernet

[SWS_APBSW_00100] Security event context data definition: SEV_ETH_DROP_UNKNOWN_ETHERTYPE ⌈

SEV Name	SEV_ETH_DROP_UNKNOWN_ETHERTYPE	
ID	15	
Description	An ethernet datagram was dropped due the Ethertype is not known.	
Context Data Version	1	
Context Data	Data Type	Allowed Values
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	EthernetFrame, truncated to the first EthIfSEvEthernetFrameMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the EthernetFrame byte array definition above.

⌋

[SWS_APBSW_00101] Security event context data definition: SEV_ETH_DROP_VLAN_DOUBLE_TAG ⌈

SEV Name	SEV_ETH_DROP_VLAN_DOUBLE_TAG	
ID	16	
Description	An ethernet datagram was dropped due to double VLAN tag.	
Context Data Version	1	
Context Data	Data Type	Allowed Values
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	EthernetFrame, truncated to the first EthIfSEvEthernetFrameMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the EthernetFrame byte array definition above.

⌋

[SWS_APBSW_00102] Security event context data definition: SEV_ETH_DROP_INV_VLAN ⌈

SEV Name	*SEV_ETH_DROP_INV_VLAN*	
ID	17	
Description	An ethernet datagram was dropped due to an invalid CrtlIdx/VLAN.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	EthernetFrame, truncated to the first EthIfSEvEthernet FrameMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the EthernetFrame byte array definition above.

⌋

[SWS_APBSW_00103] Security event context data definition: SEV_ETH_DROP_MAC_COLLISION ⌈

SEV Name	*SEV_ETH_DROP_MAC_COLLISION*	
ID	18	
Description	Ethernet datagram was dropped because local MAC was same as source MAC in an incoming frame.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	EthernetFrame, truncated to the first EthIfSEvEthernet FrameMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the EthernetFrame byte array definition above.

⌋

7.1.1.2 TCP/IP

[SWS_APBSW_00300] Security event context data definition: SEV_ARP_IP_ADDR_CONFLICT ⌈

SEV Name	*SEV_ARP_IP_ADDR_CONFLICT*	
ID	10	
Description	Received local IP address in ARP reply for different MAC.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of ReceivedPacket byte array
ReceivedPacket	uint8 [54]	Received Packet, truncated to the first TcpIpSEvReceived PacketMaxLength bytes

▽

△

SEV Name	*SEV_ARP_IP_ADDR_CONFLICT*	
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the ReceivedPacket byte array definition above.

⌋

[SWS_APBSW_00301] Security event context data definition: SEV_TCP_DROP_INV_DEST_PORT ⌈

SEV Name	*SEV_TCP_DROP_INV_DEST_PORT*	
ID	11	
Description	Dropped TCP packet because of invalid destination TCP-Port.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of ReceivedPacket byte array
ReceivedPacket	uint8 [54]	Received Packet, truncated to the first TcpIpSEvReceived PacketMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the ReceivedPacket byte array definition above.

⌋

[SWS_APBSW_00302] Security event context data definition: SEV_UDP_DROP_INV_DEST_PORT ⌈

SEV Name	*SEV_UDP_DROP_INV_DEST_PORT*	
ID	12	
Description	Dropped UDP packet because of invalid destination UDP-Port.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of ReceivedPacket byte array
ReceivedPacket	uint8 [54]	Received Packet, truncated to the first TcpIpSEvReceived PacketMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the ReceivedPacket byte array definition above.

⌋

[SWS_APBSW_00303] Security event context data definition: SEV_IPV4_DROP_INV_DEST_ADDR ⌈

SEV Name	*SEV_IPV4_DROP_INV_DEST_ADDR*	
ID	13	
Description	Dropped datagram because of invalid destination IPV4 address.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of ReceivedPacket byte array

▽

△

SEV Name	*SEV_IPV4_DROP_INV_DEST_ADDR*	
ReceivedPacket	uint8 [54]	Received Packet, truncated to the first TcpIpSEvReceived PacketMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the ReceivedPacket byte array definition above.

⌋

[SWS_APBSW_00304] Security event context data definition: SEV_IPV6_DROP_INV_DEST_ADDR ⌈

SEV Name	*SEV_IPV6_DROP_INV_DEST_ADDR*	
ID	14	
Description	Dropped datagram because of invalid destination IPV6 address.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of ReceivedPacket byte array
ReceivedPacket	uint8 [54]	Received Packet, truncated to the first TcpIpSEvReceived PacketMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the ReceivedPacket byte array definition above.

⌋

7.1.1.3 MACsec

[SWS_APBSW_00500] Security event context data definition: SEV_MKA_AUTHENTICATION_FAILURE ⌈

SEV Name	*SEV_MKA_AUTHENTICATION_FAILURE*	
ID	78	
Description	Event triggered when the authentication during the MKA communication has failed (wrong CKN/CAK).	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

[SWS_APBSW_00501] Security event context data definition: SEV_MKA_TIMEOUT ⌈

SEV Name	*SEV_MKA_TIMEOUT*	
ID	79	
Description	Event triggered when the timeout for the MKA communication has expired.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

[SWS_APBSW_00502] Security event context data definition: SEV_MKA_PORT_NOT_ENABLED ⌈

SEV Name	*SEV_MKA_PORT_NOT_ENABLED*	
ID	80	
Description	Event triggered when the indicated port for the MKA communication is not enable.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

[SWS_APBSW_00503] Security event context data definition: SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED ⌈

SEV Name	*SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED*	
ID	81	
Description	Event triggered when there is no Cipher Suite supported.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

[SWS_APBSW_00504] Security event context data definition: SEV_MKA_PORT_NUMBER_CHANGE ⌈

SEV Name	*SEV_MKA_PORT_NUMBER_CHANGE*	
ID	82	
Description	Event triggered when during the MKA communication the port number has changed.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*

▽

△

SEV Name	*SEV_MKA_PORT_NUMBER_CHANGE*	
PortId	uint8 [2]	
CKN	uint8 [32]	
MACAdressOfPeer	uint8 [6]	

⌋

7.1.1.4 Trusted Platform

[SWS_APBSW_00070] Reporting Security Event with Verification Mode set to MEASURED_BOOT to IdsM ⌈The `Base Software` shall raise the Security Event `SEV_EXEC_SW_COMPONENT_INTEGRITY_CHECK_FAILED` with Verification Mode set to MEASURED_BOOT for each affected SWComponent, if an integrity verification according to [SWS_APBSW_03507] has failed.⌋

[SWS_APBSW_00071] Reporting Security Event with Verification Mode set to STRICT to IdsM ⌈The `Base Software` shall raise the Security Event `SEV_EXEC_SW_COMPONENT_INTEGRITY_CHECK_FAILED` with Verification Mode set to STRICT for each affected SWComponent, if an integrity verification according to [SWS_APBSW_03506], [SWS_APBSW_03507], [SWS_APBSW_03508] or [SWS_APBSW_03509] has failed.⌋

[SWS_APBSW_00072] Security event context data definition: SEV_EXEC_SW_COMPONENT_INTEGRITY_CHECK_FAILED

Status: DRAFT
Previous identifiers: SWS_EM_02589

⌈

SEV Name	*SEV_EXEC_SW_COMPONENT_INTEGRITY_CHECK_FAILED*	
ID	99	
Description	The integrity check of a SW component has failed.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
SWComponent	uint16	
VerificationMode	uint8	RECOVERY (0x00) MEASURED_BOOT (0x01) RUNTIME_PERIODIC (0x02) STRICT (0x03)

⌋

7.1.2 Log Messages

The base software does not define any non-verbose log messages (i.e., modelled DLT messages).

7.1.3 Violation Messages

The base software does not define any violation messages (i.e., DLT messages logged for Violations according to [AP_SWS_GENERAL_00021]).
Please note that concrete implementations might implement Non-Standardized Violations (see also [AP_SWS_GENERAL_00003]) [5, General Specification of Adaptive Platform]

7.1.4 Production Errors

The base software does not define any production errors (i.e., Diagnostic Events).

7.2 Trusted Platform

From a security perspective, it is essential that all software executed on the Adaptive Platform is trusted, i.e. the integrity and authenticity of the software is ensured.

A key requirement for a trusted Adaptive Platform is a Trust Anchor on the `Machine` that is authentic by definition (hence that alternative name, "root of trust"). A Trust Anchor is often realized as a public key stored in a secure environment, e.g. in non-modifiable persistent memory or in an HSM. The trust has to be extended to the `Machine` by appropriate means, e.g. by a chain of trust. If the `Machine` does not exhibit a Trust Anchor, it cannot be ensured that the Adaptive Platform is trusted.

[SWS_APBSW_03500] Availability of a Trust Anchor ⌈If there is no Trust Anchor available on the `Machine`, the following requirements may be ignored: [SWS_APBSW_03501], [SWS_APBSW_03502], [SWS_APBSW_03503], [SWS_APBSW_03504], [SWS_APBSW_03505], [SWS_APBSW_03506], [SWS_APBSW_03507], [SWS_APBSW_03508], [SWS_APBSW_03509], [SWS_APBSW_03510].⌋

There are many ways to verify the integrity and authenticity of the Adaptive Platform. A `Trusted Platform` can be realized e.g. (but not limited to) by

- Verification of the complete Ramdisk by the Bootloader
- Verification of individual `Executable` and data files, e.g. using OS-functionalities or a trusted third-party process
- Verification of individual memory pages upon being loaded, e.g. using OS-functionalities or a trusted third-party process

[SWS_APBSW_03501] Integrity and Authenticity of Machine configuration ⌈The `Base Software` shall ensure that the integrity and authenticity of Machine information from the `Processed Manifest` are checked before use.⌋

[SWS_APBSW_03502] Integrity and Authenticity of each `Executable` ⌈The `Base Software` shall ensure that for every `Process` that is about to be started, the integrity and authenticity of the `Executable` itself are checked.⌋

[SWS_APBSW_03503] Integrity and Authenticity of shared objects ⌈The `Base Software` shall ensure that for every `Process` that is about to be started, the integrity and authenticity of each related shared object are checked.⌋

[SWS_APBSW_03504] Integrity and Authenticity of Processed Manifest configurations ⌈The `Base Software` shall ensure that for every `Process` that is about to be started, the integrity and authenticity of its corresponding `Processed Manifest` are checked.⌋

The information validated by [SWS_APBSW_03504] includes all manifest information, e.g. Service Instance information, and not just the information directly used by `Execution Management`.

From a security perspective, the rationale for choosing these items is as follows:

- `Executable`: Modifying the `Executable` itself allows an attacker to execute arbitrary code on the `Machine`;
- Manifests: `Machine Manifest`, `Execution Manifest` and `Service Instance Manifest` describe what and how something should be executed and are thus an obvious attack vector on the Adaptive Platform;
- Shared Objects: Shared objects can either contain code that is executed within the context of the `Process` or data that (potentially) influences the execution of a `Process` accessing this data. A modified shared object could consequently be used to compromise the system.

In order to establish a `Trusted Platform`, it must be ensured that only trusted software is launched. Therefore, a system designer has to ensure that the `Machine.trustedPlatformExecutableLaunchBehavior` is started authentically. For instance, this could be realized by a chain of trust as described in [6].

The `Base Software` in turn has to ensure that all `Executable` code on the Adaptive Platform is authenticated before being executed. An example of a complete authenticated start-up sequence can look like this:

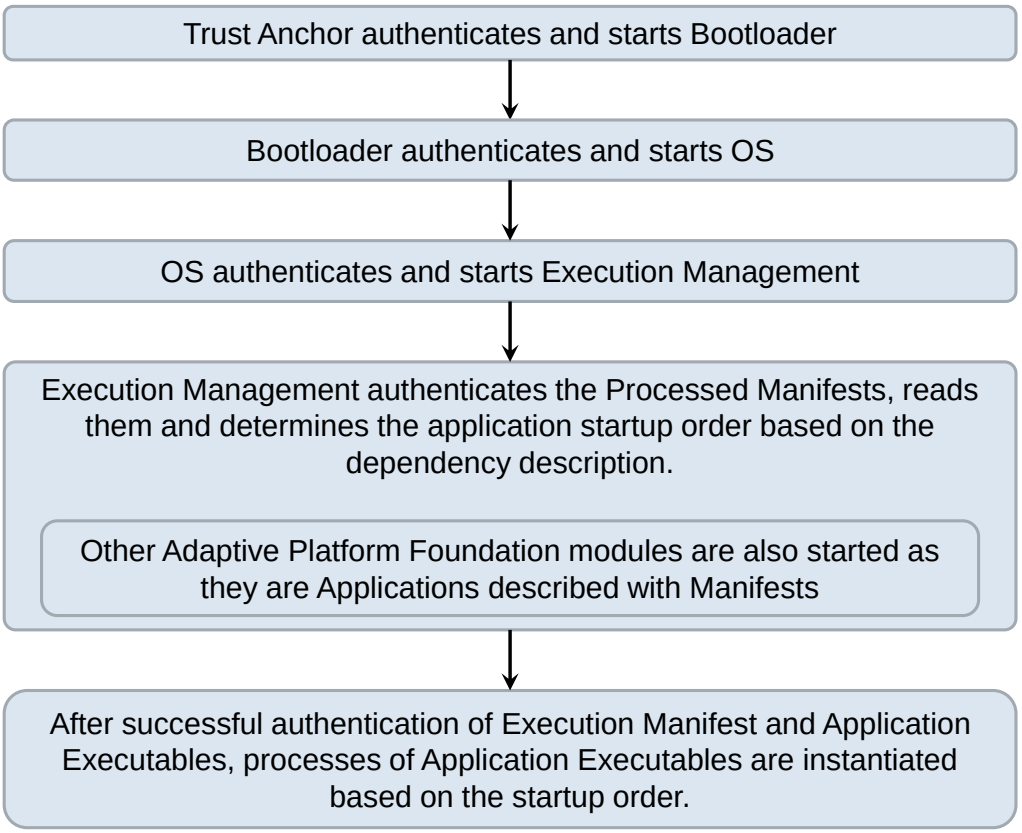


Figure 7.1: Authenticated start-up sequence

The integrity and authenticity of persistent data stored by applications is not considered here. The `Functional Cluster` Persistency takes care of the integrity of this data.

7.2.1 Handling of failed authenticity checks

If the integrity and authenticity has been verified successfully, the system has to continue with its regular start-up process. If the integrity and authenticity check has failed, however, the `Machine.trustedPlatformExecutableLaunchBehavior` offers a configuration option on how to proceed with the start-up process.

[SWS_APBSW_03505] Failed authenticity checks ⌈The `Base Software` shall select the trusted platform mode based on the value of `Machine.trustedPlatformExecutableLaunchBehavior`.⌋

The configuration of the three modes is done via the `trustedPlatformExecutableLaunchBehavior` attribute within the `Processed Manifest`. The configuration option is only allowed to be processed after the integrity and authenticity of the relevant `Processed Manifest` has been verified.

These three modes of the `Machine.trustedPlatformExecutableLaunchBehavior` are:

- monitorMode
- noTrustedPlatformSupport
- strictMode

[SWS_APBSW_03506] Launch Behavior Validation ⌈The Base Software shall stop the start-up sequence of the Adaptive Platform if the integrity or authenticity check of the Processed Manifest containing the trustedPlatformExecutableLaunchBehavior selection has failed and switch to the unrecoverable state.⌋

The integrity and authenticity check applies to all trusted platform modes; to do otherwise would leave the system open to attacks that maliciously corrupt the Manifest information. Reaction to a failure is limited as, by definition, no Adaptive Application other than Execution Management are running and hence are restricted to implementation defined actions such as OS-level logging.

7.2.1.1 Monitor Mode

In monitorMode, the integrity and authenticity checks are performed, but the start-up process is not affected. Hence, the Adaptive Platform starts up even if the file system has been compromised.

[SWS_APBSW_03507] Monitor Mode ⌈In monitorMode, if a check ([SWS_APBSW_03501], [SWS_APBSW_03502], [SWS_APBSW_03503] and [SWS_APBSW_03504]) fails then the Base Software shall log the failure as DltMessage FailedIntegrityOrAuthenticityCheck and continue regular startup of the Adaptive Platform.⌋

monitorMode is useful when the integrator wants the system to keep running, even if the platform is not considered trusted. In this case, the integrator might use additional measures outside the scope of Adaptive AUTOSAR, like e.g. restricted key access when using an HSM that supports this feature.

monitorMode is also useful during development phase, when frequent changes on the Adaptive Platform are performed and keeping the authentication tag (e.g. signatures) valid is a tedious task.

7.2.1.2 Strict Mode

In strictMode, the Adaptive Platform ensures that no Process are executed, where the integrity and authenticity of the corresponding Executable, manifests or linked library could not be verified.

[SWS_APBSW_03508] Strict Mode - Execution manifest ⌈In strictMode, the Base Software shall prevent the execution of an Executable if the integrity or authenticity check of the corresponding processed Execution Manifest has failed.⌋

[SWS_APBSW_03509] Strict Mode - Service Instance manifests ⌈In strictMode, the Base Software shall prevent the execution of an Executable if the integrity or authenticity check of at least one of the corresponding processed Service Instance Manifest has failed.⌋

[SWS_APBSW_03510] Strict Mode - Executables ⌈In strictMode, the Base Software shall allow to start a Process only if the integrity and authenticity of the corresponding Executable and all shared objects linked with it was successfully verified.⌋

Executable code can be provided by executables and by statically linked shared objects linked by the executable. The Base Software cannot determine dynamically linked shared objects and thus these needs to be validated through an alternative, implementation specific, mechanism.

Example: Consider an Adaptive Platform in strictMode. Execution Management has started several Executable after the Base Software has successfully ensured the integrity and authenticity of the Executable, its related shared objects and its Processed Manifest. Now, Execution Management wants to start another Executable, where the authenticity check has failed. Execution Managementcannot launch this Executable, because the Base Software rendered this Executable unavailable by project or platform specific measures (e.g., unmounting a broken partition). The other Executable that passed the authenticity check may however continue to run. When Execution Managementattempts to start another Executable it can be started as long as all authenticity checks are passed.

7.3 MACsec

MACsec is a secure communication protocol on MAC layer, as defined in IEEE-802.1X and IEEE802.1AE. The secure communication is performed by the Security Entitiy (SecY), which is typically implemented in the PHY. The SecYs are managed by the MACsec Key Agreement module, which is typically implemented on the level of the base software (see [7, MACsec explanatory document] for more details).

[SWS_APBSW_03000] Base SW MACsec support

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00006

⌈The Base Software shall support MACsec and MKA according to the [8, Requirements on MACsec]⌋

[SWS_APBSW_03001] MACsec configuration

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00003, FO_RS_MACsec_00004, FO_RS_MACsec_00007, FO_RS_MACsec_00008

⌈The MKA in the Base Software shall be configured as defined in MacSecLocalKayProps.⌋

[SWS_APBSW_03002] Daemon start-up

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00005

⌈The Base SW shall initialize the MKA Daemon upon start-up, so that MACsec secure channels are established.⌋

[SWS_APBSW_03003] Link indication

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00003

⌈The Base Software shall indicate link-up and link-down to MKA.⌋

7.4 Demands and constraints for ARTI tracing

The `ARTI` tracing interface is used to understand, verify and visualize the timing behavior of the OS. It is used to collect information about `Task` and `Process` of the OS.

The `ARTI` interface follows the two-level approach of AUTOSAR, where a "`Task`" is a schedulable unit (in POSIX and C++ called "thread"), and a "`Process`" is a mandatory environment holding several `Task`. A system may look like this:

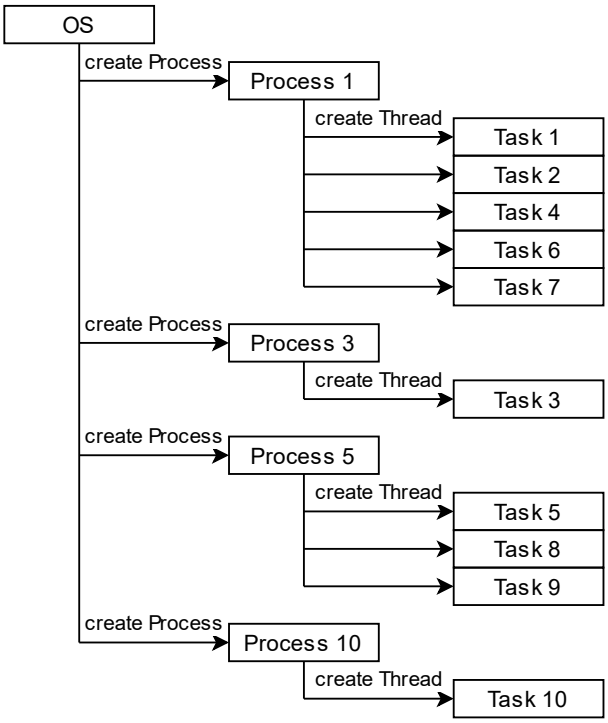


Figure 7.2: Process - Task/Thread Model

The OS provides the information of `Process` and `Task` in different ways. It can implement trace buffers that contain kernel internal information, it could provide proprietary hooks from the kernel with internal information or it could provide modeled messages out of the box.

These different variants of the OS specific information need to be sent using modeled messages to the ara::log API. This can be implemented in different ways.

For example, for `Operating System` that are using trace buffers with internal information, there could be a separate application. Here such application or daemon is called “OS/ara::log Adapter”. The Figure 7.3 “Example layout of the OS/ara::log Adapter” shows how it integrates in the AUTOSAR framework. The OS/ara::log Adapter reads OS specific trace buffers, translates the information to modeled messages and sends them to the ara::log API.

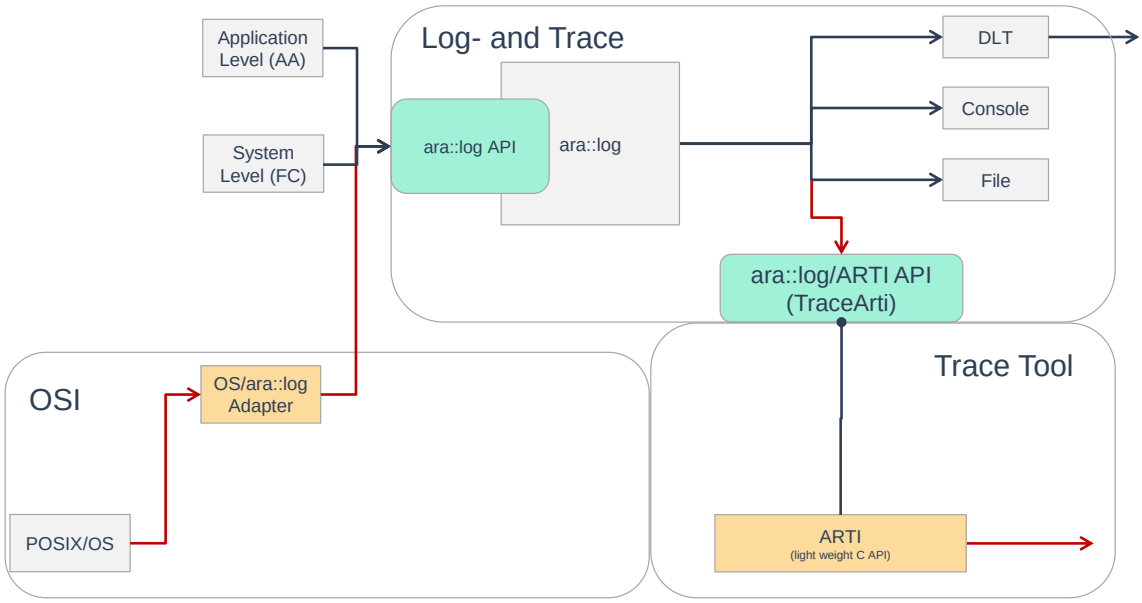


Figure 7.3: Example layout of the OS/ara::log Adapter

Sending the modeled messages can also be delegated to a different `Functional Cluster`. This can be helpfull when sending these messages would be the only active part of the `Operating System Interface`.

The initial state of existing `Process` and `Task` when the tracing is started is logged using the OsProcessInfo and OsTaskInfo message. This initial state assures that `Task` ids and `Process` ids can be correctly interpreted and can be assigned to `Executable`.

[SWS_APBSW_02000] Log OS tracing started ⌈Whenever the `Operating System Interface` starts the tracing of `Process` and `Task`, it shall

- log a modeled message of type `OsProcessInfo` for each `Process` currently available.
- log a modeled message of type `OsTaskInfo` for each `Task` currently available.

⌋

While the concepts of `Process` and `Task` are common in most memory-partitioning OSes, there are still significant variations on how these concepts are concretely implemented. As a consequence, the order of declaration of creation, destruction and modification of each resource logged in the output is just weakly defined. The only requirement is that the log describes a coherent view of the system. Tooling must be tolerant to the variety of ordering while decoding the information.

7.4.1 Tracing of tasks

The term `Task` applies to the object as defined in the AUTOSAR Glossary: “A `Task` is the smallest schedulable unit managed by the OS. The OS decides when and which task can run on the core(s) of the ECU.”

The trace events of a `Task` shall follow the state machine in Figure 7.4.

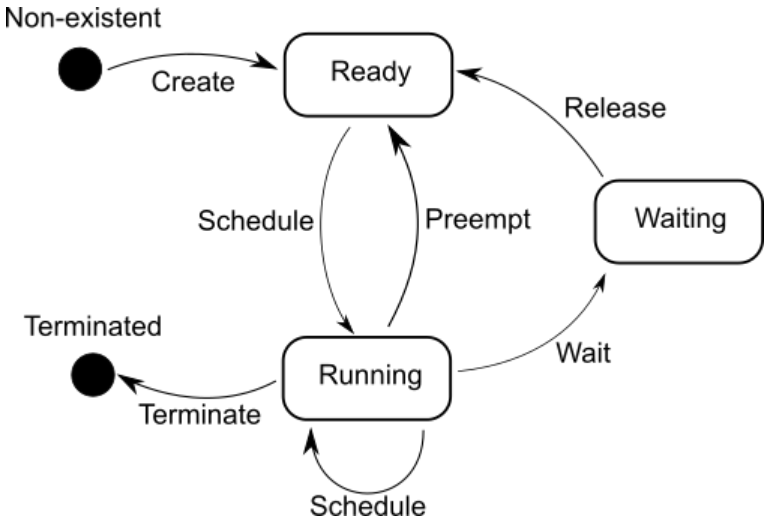


Figure 7.4: minimal state machine of a task

The minimal state machine for a single `Task` has the states:

Ready - The `Task` is ready and can be scheduled for running

Running - The `Task` is being executed

Waiting - The `Task` is waiting for an event, semaphore, a different thread or different OS object. The `Task` can not be scheduled for running.

For an OS that does not support or differentiate between Ready state and Waiting state, the `ARTI` trace events for tracing switches between Ready and Running shall be mandatory, and `ARTI` trace events for switching to Waiting state are optional.

The trace points that are related to `Task` are:

[SWS_APBSW_02001] Log Task Schedule Notification ⌈Whenever an OS `Task` is scheduled and is entering the running state, the `Operating System Interface` shall log a modeled message of type `OsTaskSchedule`.⌋

[SWS_APBSW_02002] Log Task Wait Notification ⌈Whenever an OS `Task` enters the wait state, the `Operating System Interface` shall log a modeled message of type `OsTaskWait`.⌋

[SWS_APBSW_02003] Log Task Release Notification ⌈Whenever an OS `Task` is released, the `Operating System Interface` shall log a modeled message of type `OsTaskRelease`.⌋

[SWS_APBSW_02004] Log Task Preempt Notification ⌈Whenever an OS `Task` is preempted, the `Operating System Interface` shall log a modeled message of type `OsTaskPreempt`.⌋

[SWS_APBSW_02005] Log Task Exit Notification ⌈Whenever an OS `Task` exits, the `Operating System Interface` shall log a modeled message of type `OsTaskTerminate`.⌋

[SWS_APBSW_02006] Log Task Creation Notification ⌈Whenever an OS `Task` is created, the `Operating System Interface` shall log a modeled message of type `OsTaskCreate`.⌋

[SWS_APBSW_02007] Log Task Renaming Notification ⌈Whenever an OS `Task` is renamed, the `Operating System Interface` shall log a modeled message of type `OsTaskRename`.⌋

The parameter `eventTime` shall cover the time when the event occurred. This assures the most accurate time that is possible. The format of that timestamp is the natural format of the OS.

Log messages shall be provided for the following task related OS events:

[SWS_APBSW_02030] Log Message OsTaskCreate

Status: DRAFT
Previous identifiers: SWS_OSI_02008

⌈

Dlt-Message	OsTaskCreate		
Description	Notify the tracer about the creation of a task		
MessageId	0x8000e005		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
coreId	Id of the core	uint32 [1]	NoUnit
posixProcessId	OS specific PID which has been assigned to the process the task is in	uint32 [1]	NoUnit
taskId	Id of the task that is created	uint32 [1]	NoUnit

⌋

[SWS_APBSW_02031] Log Message OsTaskTerminate

Status: DRAFT
Previous identifiers: SWS_OSI_02009

⌈

Dlt-Message	OsTaskTerminate		
Description	Notify the tracer about the exit of a task		
MessageId	0x8000e006		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
coreId	Id of the core	uint32 [1]	NoUnit
taskId	Id of the task that exits	uint32 [1]	NoUnit

⌋

[SWS_APBSW_02032] Log Message OsTaskInfo

Status: DRAFT
Previous identifiers: SWS_OSI_02010

⌈

Dlt-Message	OsTaskInfo		
Description	Provide information of an existing task		
MessageId	0x8000e007		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
taskId	Id of the task	uint32 [1]	NoUnit
posixProcessId	OS specific PID which has been assigned to the parent process	uint32 [1]	NoUnit
taskName	New name of the task	uint8 [32, encoding UTF-8]	

⌋

[SWS_APBSW_02033] Log Message OsTaskPreempt

Status: DRAFT
Previous identifiers: SWS_OSI_02011

⌈

Dlt-Message	OsTaskPreempt		
Description	Notify the tracer that a task is leaving running state		
MessageId	0x8000e008		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***

▽

△

eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
coreId	Id of the core	uint32 [1]	NoUnit
taskId	Id of the task that is leaving the running state and enters the ready state	uint32 [1]	NoUnit

⌋

[SWS_APBSW_02034] Log Message OsTaskRelease

Status: DRAFT
Previous identifiers: SWS_OSI_02012

⌈

Dlt-Message	OsTaskRelease		
Description	Notify the tracer that a task is leaving the wait state		
MessageId	0x8000e009		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
coreId	Id of the core	uint32 [1]	NoUnit
taskId	Id of the task that is leaving the wait state	uint32 [1]	NoUnit

⌋

[SWS_APBSW_02035] Log Message OsTaskRename

Status: DRAFT
Previous identifiers: SWS_OSI_02013

⌈

Dlt-Message	OsTaskRename		
Description	Provide a name for a task		
MessageId	0x8000e00a		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
taskId	Id of the task	uint32 [1]	NoUnit
taskName	New name of the task	uint8 [32, encoding UTF-8]	

⌋

[SWS_APBSW_02036] Log Message OsTaskSchedule

Status: DRAFT
Previous identifiers: SWS_OSI_02014

⌈

Dlt-Message	OsTaskSchedule		
Description	Notify the tracer about that the task is scheduled and is entering the running state		
MessageId	0x8000e00b		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
coreId	Id of the core	uint32 [1]	NoUnit
nextTaskId	Id of the task that starts running	uint32 [1]	NoUnit

⌋

[SWS_APBSW_02037] Log Message OsTaskWait

Status: DRAFT
Previous identifiers: SWS_OSI_02015

⌈

Dlt-Message	OsTaskWait		
Description	Notify the tracer that a task is entering the wait state		
MessageId	0x8000e00c		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
eventTime	OS implementation specific time when the event occurred in the OS.	uint64	NoUnit
coreId	Id of the core	uint32	NoUnit
taskId	Id of the task that is entering the wait state	uint32	NoUnit

⌋

7.4.2 Tracing of processes

The term `Process` applies to the object as defined in the AUTOSAR Glossary: “An `Executable` unit managed by an `Operating System` scheduler that has its own name space and resources (including memory) protected against the use by other `Process`.”

The trace points that are related to `Process` are:

[SWS_APBSW_02040] Log Process Switch Notification ⌈Whenever an OS `Process` is switched, the `Operating System Interface` shall log a modeled message of type `OsProcessSwitch`.⌋

[SWS_APBSW_02041] Log Process Creation Notification ⌈Whenever an OS `Process` is created, the `Operating System Interface` shall log a modeled message of type `OsProcessCreate`.⌋

[SWS_APBSW_02042] Log Process Ending Notification ⌈Whenever an OS `Process` ends, the `Operating System Interface` shall log a modeled message of type `OsProcessDestroy`.⌋

[SWS_APBSW_02043] Log Process Renaming Notification ⌈Whenever an OS `Process` is renamed, the `Operating System Interface` shall log a modeled message of type `OsProcessRename`.⌋

The parameter `eventTime` shall cover the time when the event occurred. This assures the most accurate time that is possible. The format of that timestamp is the natural format of the OS.

Log messages shall be provided for the following process related OS events:

[SWS_APBSW_02050] Log Message OsProcessCreate

Status: DRAFT
Previous identifiers: SWS_OSI_02003

⌈

Dlt-Message	OsProcessCreate		
Description	Notify the tracer about the creation of a process		
MessageId	0x8000e000		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
coreId	Id of the core	uint32 [1]	NoUnit
posixProcessId	OS specific PID which has been being created	uint32 [1]	NoUnit
posixParent ProcessId	OS specific PID which has been assigned to the parent process	uint32 [1]	NoUnit

⌋

[SWS_APBSW_02051] Log Message OsProcessDestroy

Status: DRAFT
Previous identifiers: SWS_OSI_02004

⌈

Dlt-Message	OsProcessDestroy		
Description	Notify the tracer about the destruction of a process		
MessageId	0x8000e001		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***

▽

△

eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
coreId	Id of the core	uint32 [1]	NoUnit
posixProcessId	OS specific PID which has been assigned to the process that is being destroyed	uint32 [1]	NoUnit

⌋

[SWS_APBSW_02052] Log Message OsProcessInfo

Status: DRAFT
Previous identifiers: SWS_OSI_02005

⌈

DIt-Message	OsProcessInfo		
Description	Provide information of an existing process		
MessageId	0x8000e002		
MessageType Info	DLT_TRACE_STATE		
DIt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
posixProcessId	OS specific PID which has been assigned to the process	uint32 [1]	NoUnit
posixParent ProcessId	OS specific PID which has been assigned to the parent process	uint32 [1]	NoUnit
posixProcess Name	OS specific name of the process that is specified by posixProcessId.	uint8 [32, encoding UTF-8]	

⌋

[SWS_APBSW_02053] Log Message OsProcessRename

Status: DRAFT
Previous identifiers: SWS_OSI_02006

⌈

DIt-Message	OsProcessRename		
Description	Provide a name for a process		
MessageId	0x8000e003		
MessageType Info	DLT_TRACE_STATE		
DIt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
posixProcessId	OS specific PID which has been assigned to the process	uint32 [1]	NoUnit
posixProcess Name	OS specific name of the process that is specified by posixProcessId.	uint8 [32, encoding UTF-8]	

⌋

[SWS_APBSW_02054] Log Message OsProcessSwitch

Status: DRAFT
Previous identifiers: SWS_OSI_02007

⌈

Dlt-Message	OsProcessSwitch		
Description	Notify the tracer about the switch of a process		
MessageId	0x8000e004		
MessageType Info	DLT_TRACE_STATE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
eventTime	OS implementation specific time when the event occurred in the OS.	uint64 [1]	NoUnit
coreId	Id of the core	uint32 [1]	NoUnit
posixProcessId	OS specific PID which has been assigned to the process	uint32 [1]	NoUnit

⌋

8 API Specification

This specification does not specify a functional cluster. This chapter does not apply.

9 Service Interfaces

This specification does not specify a functional cluster. This chapter does not apply.

10 Configuration

Aspects of configuration of Specification of Base Software are described in [9]. This chapter defines default values and semantic constraints for this configuration model.

10.1 Default Values

This section defines default values for lower layer software (not related to an individual functional cluster) where useful for selected configuration parameters (where provision of such default value seems to be straightforward). These are defined in [9]. The motivation for providing these values is to simplify setting up an adaptive development project.

In other words the default values are meant to guide certain roles (e.g. developer, integrator) which values to choose to quickly get to a working configuration for Adaptive Machines. This document lists default values for parameters, which were proposed by AUTOSAR partners, based on experience from development projects. By intention, it makes no distinction between design and deployment.

Some default values are taken over from [10]. When there is no equivalent note provided, a default value is specified by AUTOSAR partners. It is worth noting that in any case the parameters and their default settings have been chosen for the optimization of in-vehicle communication between a AUTOSAR adaptive or classic platform instance with some vehicle local communication peer. This means optimization of TCP/IP communication for Automotive Ethernet. Hence the default values should work as well for in-vehicle communication including at least one adaptive platform communication peer.

External vehicle communication has not been considered for the definition of default values. Other types of physical media (e.g. for wireless connectivity) may require other TCP/IP values for optimizing communication. For example communication over media with longer round trip times (compared to Ethernet) need larger values for timeout settings. These aspects need to be taken into consideration based on the expected mixture of traffic over diverse type of media.

10.1.1 Default values for TCP parameters

Qualified attribute name	Default Value	Description
MachineDesign.tcpIpProps.tcpProps.tcpDelayedAckTimeout	0.01	maximum time in seconds an acknowledgment is delayed for transmission
MachineDesign.tcpIpProps.tcpProps.tcpFinWait2Timeout	1	timeout in seconds to receive a FIN from the remote node, after the local node has initiated connection termination, i.e. maximum time waiting in FINWAIT-2 state for a connection termination request from the remote TCP.



△

MachineDesign.tcpIpProps.tcpProps.tcpKeepAliveEnabled	TRUE	Keep alive is assumed to be an important mechanism to detect potential problems with communication peer. If enabled both peers regularly exchange messages to verify the connection status.
MachineDesign.tcpIpProps.tcpProps.tcpKeepAliveInterval	1	interval in seconds between subsequent keepalive probes
MachineDesign.tcpIpProps.tcpProps.tcpKeepAliveProbesMax	5	maximum number of times that a TCP Keep Alive is retransmitted before the connection is closed
MachineDesign.tcpIpProps.tcpProps.tcpKeepAliveTime	5	time in seconds between the last data packet sent (simple ACKs are not considered data) and the first keepalive probe
MachineDesign.tcpIpProps.tcpProps.tcpMaxRtx	5	maximum number of times that a TCP segment is retransmitted before the TCP connection is closed. This parameter is only valid if tcpSynMaxRtx is configured. Note: This parameter also applies for FIN retransmissions.
MachineDesign.tcpIpProps.tcpProps.tcpMsl	in the range between 0.5 and 30	maximum segment lifetime in seconds
MachineDesign.tcpIpProps.tcpProps.tcpNagleEnabled	FALSE	specifies whether Nagle algorithm is enabled. The algorithm improves efficiency of TCP/IP networks by reducing the number of packets that need to be sent over the network.
MachineDesign.tcpIpProps.tcpProps.tcpReceiveWindowMax	8192 (8 KByte)	maximum size of TCP window in byte
MachineDesign.tcpIpProps.tcpProps.tcpRetransmissionTimeout	in the range between 0.01 and 0.5	timeout in seconds before an unacknowledged TCP segment is sent again. If the timeout is disabled, no TCP segments shall be retransmitted.
MachineDesign.tcpIpProps.tcpProps.tcpSynMaxRtx	5	maximum number of times that a TCP SYN is retransmitted. SYN message is sent to request a TCP connection setup.
MachineDesign.tcpIpProps.tcpProps.tcpTtl	255	time-to-live in number of hops for outgoing TCP packets. This mechanism prevents undelivered TCP segments to be forwarded between nodes for an unspecified number of hops.

10.1.2 Default values for UDP parameters

Qualified attribute name	Default Value	Description
MachineDesign.tcpIpProps.udpProps.udpTtl	255	time-to-live in number of hops for outgoing UDP packets.This mechanism prevents undelivered UDP segments to be forwarded between nodes for an unspecified number of hops.

10.1.3 Default values for ICMPv4 parameters

Qualified attribute name	Default Value	Description
MachineDesign.tcpIpIcmpProps.icmpV4Props.tcpIpIcmpV4EchoReplyEnabled	FALSE	enables or disables transmission of ICMPv4 messages. Value is taken from classic platform [10]. Further it is assumed that there are more sophisticated mechanisms for diagnostic monitoring.

10.1.4 Default values for ICMPv6 parameters

Qualified attribute name	Default Value	Description
MachineDesign.tcpIpIcmpProps.icmpV6Props.tcpIpIcmpV6EchoReplyAvoidFragmentation	TRUE	Defines whether the echo reply is only transmitted in case that the incoming ICMPv6 Echo Request (Pings) fits the MTU of the respective interface, i.e. can be transmitted without IPv6 fragmentation. Assumption is that fragmentation of ICMP echo request will be no issue inside vehicle network (based on Ethernet communication).
MachineDesign.tcpIpIcmpProps.icmpV6Props.tcpIpIcmpV6EchoReplyEnabled	TRUE	Parameter enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception.
MachineDesign.tcpIpIcmpProps.icmpV6Props.tcpIpIcmpV6HopLimit	255	time-to-live in number of hops for outgoing ICMPv6 packets. This mechanism prevents undelivered ICMPv6 packets to be forwarded between nodes for an unspecified number of hops.
MachineDesign.tcpIpIcmpProps.icmpV6Props.tcpIpIcmpV6MsgDestinationUnreachableEnabled	TRUE	Parameter enables or disables transmission of ICMP Destination Unreachable Messages.
MachineDesign.tcpIpIcmpProps.icmpV6Props.tcpIpIcmpV6MsgParameterProblemEnabled	TRUE	If enabled an ICMPv6 parameter problem message will be sent if a received packet has been dropped due to unknown options or headers that are found in the packet.

10.1.5 Default values for ARP parameters

Qualified attribute name	Default Value	Description
MachineDesign.ethIpProps.arpProps.tcpIpArpNumGratuitousArpOnStartup	0	number of gratuitous ARP replies which shall be sent on assignment of a new IP address. This is a specific type of ARP response message which was not triggered by an ARP request message.
MachineDesign.ethIpProps.arpProps.tcpIpArpPacketQueueEnabled	FALSE	Enables or disables support of the ARP packet queue according to IETF RFC 1122. Value is taken from classic platform [10].

▽

△

MachineDesign.ethIpProps.arp-Props.tcpIpArpRequestTimeout	1	Timeout in seconds for the validity of ARP requests. After the transmission of an ARP request further transmission of ARP requests to the same destination shall be skipped for the specified duration. Value is taken from classic platform [10].

10.1.6 Default values for AutoIP parameters

Qualified attribute name	Default Value	Description
MachineDesign.ethIpProps.autoIp-Props.tcpIpAutoIpInitTimeout	0	Specifies the time in seconds Auto-IP waits at startup, before beginning with ARP probing. This delay is used to give DHCP time to acquire a lease in case a DHCP server is present.

10.1.7 Default values for IPv4 fragmentation parameters

Qualified attribute name	Default Value	Description
MachineDesign. ethIpProps.fragmentationProps.tcp-IpIpFragmentationRxEnabled	FALSE	Enables or disables support for reassembling of incoming datagrams that are fragmented according to IETF RFC 815 (IP Datagram ReassemblyAlgorithms). At least for in-vehicle communication with nodes connected to MTU of same size, Ethernet fragmentation is not needed. In case this argumentation is followed subsequent fragmentation related parameters, all of them taking the following prefix MachineDesign. ethIpProps.fragmentationProps. have no role in setting default values: tcpIpIpNumFragments tcpIpIpNumReassDgrams tcpIpIpReassTimeout

10.1.8 Default values for DHCPv6 parameters

Qualified attribute name	Default Value	Description
MachineDesign.ethIpProps.dhcp-Props.tcpIpDhcpV6CnfDelayMax	1	maximum delay in seconds before sending the first confirm message. Value is taken from classic platform [10].
MachineDesign.ethIpProps.dhcp-Props.tcpIpDhcpV6CnfDelayMin	0	minimum delay in seconds before the first confirm message will be sent. Value is taken from classic platform [10].

▽

△

MachineDesign.ethIpProps.dhcpProps.tcpIpDhcpV6InfDelayMax	1	maximum delay in seconds before sending the first information request message. Value is taken from classic platform [10].
MachineDesign.ethIpProps.dhcpProps.tcpIpDhcpV6InfDelayMin	0	minimum delay in seconds before sending the first information request message. Value is taken from classic platform [10].
MachineDesign.ethIpProps.dhcpProps.tcpIpDhcpV6SolDelayMax	1	maximum delay in seconds before sending the first solicit message. Value is taken from classic platform [10].
MachineDesign.ethIpProps.dhcpProps.tcpIpDhcpV6SolDelayMin	0	minimum delay in seconds before the first solicit message will be sent. Value is taken from classic platform [10].

10.1.9 Default values for NDP parameters

Qualified attribute name	Default Value	Description
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDefaultReachableTime	120	reachable time for NDP in seconds as specified by IETF RFC 4861.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDefaultRetransTimer	in the range between 0.1 and 1	retransmission timer for NDP in seconds as specified by IETF RFC 4861.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDefaultRouterListSize	2	size of router list. These entries may be used to send NDP messages. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDefensiveProcessing	FALSE	If enabled the NDP shall only process Neighbor Advertisements which are received in reaction to a previously transmitted Neighbor Solicitation as well as skipping updates to the Neighbor Cache based on received Neighbor Solicitations. If disabled all Neighbor Advertisements and Solicitations shall be processed as specified in RFC4861.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDelayFirstProbeTimeValue	5	delay in seconds before sending the first "Neighbor Unreachable Detection" (NUD) probe. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDestinationCacheSize	5	maximum number of entries in the destination cache, which contains entries about all hosts on the network, which have already sent packets. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDynamicHopLimitEnabled	FALSE	If enabled the default hop limit may be reconfigured based on received router advertisements.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDynamicMtuEnabled	FALSE	If enabled allows dynamic reconfiguration of link MTU via router advertisements.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDynamicReachableTimeEnabled	TRUE	If enabled the default reachable time value may be reconfigured based on received router advertisements.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpDynamicRetransTimeEnabled	TRUE	If enabled the default retransmit timer value may be reconfigured based on received router advertisements.

▽

△

MachineDesign.ethIpProps.ndpProps.tcpIpNdpMaxRandomFactor	15	maximum random factor used for randomization. In randomized neighbor discovery, each node transmits at randomly chosen times. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpMaxRtrSolicitationDelay	1	maximum delay before the first Router Solicitation will be sent after interface initialization in seconds. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpMaxRtrSolicitations	1	maximum number of Router Solicitations that will be sent before the first Router Advertisement has been received. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpMinRandomFactor	5	minimum random factor used for randomization. In randomized neighbor discovery, each node transmits at randomly chosentimes. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpNeighborUnreachabilityDetectionEnabled	TRUE	If enabled Neighbor Unreachability Detection is used to remove unused entries from the neighbor cache. This feature is a basic feature of NDP and should be turned on.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpNumMulticastSolicitations	3	maximum number of multicast solicitations that will be sent when performing address resolution. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpNumUnicastSolicitations	3	maximum number of unicast solicitations that will be sent when performing Neighbor Unreachability Detection. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpPacketQueueEnabled	TRUE	If enabled supports a NDP Packet Queue according to IETF RFC 4861.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpPrefixListSize	5	maximum number of entries in the on-link prefix list. List includes valid address prefixes on the link, which may be used to protect the network from invalid messages. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpRandomReachableTimeEnabled	TRUE	If enabled the value of ReachableTime will be multiplied with a random value between minimum and maximum random factor in order to prevent multiple nodes from transmitting at exactly the same time.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpRndRtrSolicitationDelayEnabled	TRUE	If enabled the first router solicitation will be delayed randomly between 0 and the maximum router solicitation delay. Otherwise the first router solicitation will be sent after exactly the value of specified maximum router solicitation delay. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpRtrSolicitationInterval	4	interval between consecutive Router Solicitations in seconds. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpSlaacDadNumberOfTransmissions	1	number of Neighbor Solicitations that have to be unanswered in order to set an autoconfigured address to "preferred"" (usable) state. Value is taken from classic platform [10].
MachineDesign.ethIpProps.ndpProps.tcpIpNdpSlaacDadRetransmissionDelay	1	maximum value for the address configuration delay seconds. Value is taken from classic platform [10].

▽

△

MachineDesign.ethIpProps.ndpProps.tcpIpNdpSlaacDelayEnabled	FALSE	If enabled transmission of the first DAD (duplicate address detection). Neighbor Solicitation will be delayed by a random value between 0 and maximum DAD delay.
MachineDesign.ethIpProps.ndpProps.tcpIpNdpSlaacOptimisticDadEnabled	FALSE	If enabled it represents Optimistic Duplicate Address Detection (DAD)according to RFC4429 Value is taken from classic platform [10].

10.1.10 Default values for MACsec parameters

Qualified attribute name	Default Value	Description
MachineDesign.communicationController.couplingPort.macSecProps.macSecKayConfig.destinationMacAddress	01-80-C2-00-00-03	This attribute defines the multicast destination MAC Address that is used to send MKA PDUs
MachineDesign.communicationController.couplingPort.macSecProps.macSecKayConfig.keyServerPriority	0 for KeyServer, 255 for Peer	This attribute defines the key-server priority.
MachineDesign.communicationController.couplingPort.macSecProps.macSecKayConfig.role	MKA_KEY_SERVER for key server or MKA_PEER for peer	Role of the MAC Security Key Agreement Entity
MachineDesign.communicationController.couplingPort.macSecProps.autoStart	TRUE	This attribute defines how the Port Access Entity (PAE) is started:_ true := Autostart_ false := Manual Start
MachineDesign.communicationController.couplingPort.macSecProps.onFailPermissiveMode	MKA_PERMISSIVE_MODE_NEVER	This attribute sets the behavior of the Port Access Entity in case MACsec does not succeed
MachineDesign.communicationController.couplingPort.macSecProps.onFailPermissiveModeTimeout	3	Timeout in seconds to enable the controlled port in case onFailPermissiveMode is set to Timeout.
MachineDesign.communicationController.couplingPort.macSecProps.sakRekeyTimeSpan	0	Time in seconds to trigger the rekey of an in use SAK (Static Secure Association key). If set to 0, the rekey will not be triggered after a time span.

10.2 Semantic Constraints

This specification does not specify a functional cluster. This chapter does not apply.

A Mentioned Manifest Elements

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Chapter is generated.

Class	**CouplingPort**			
Note	A CouplingPort is used to connect a CouplingElement with an EcuInstance or two CouplingElements with each other via a CouplingPortConnection. Optionally, the CouplingPort may also have a reference to a macMulticastGroup and a defaultVLAN.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CouplingElement.couplingPort, EthernetCommunicationController.couplingPort			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
connection Negotiation Behavior	EthernetConnection NegotiationEnum	0..1	attr	Specifies the connection negotiation of the CouplingPort.
couplingPort Details	CouplingPortDetails	0..1	aggr	Defines more details of a CouplingPort in case a more specific configuration is required. **Tags:** atp.Splitkey=couplingPortDetails
couplingPort Role	CouplingPortRoleEnum	0..1	attr	Defines the role this CouplingPort takes in the context of the CouplingElement.
defaultVlan	EthernetPhysical Channel	0..1	ref	The vLanIdentifier of the referenced VLAN is the Default-PVID (port VLAN ID). A Port VLAN ID is a default VLAN ID that is assigned to an access CouplingPort to designate the VLAN segment to which this port is connected. Also, if a CouplingPort has not been configured with any VLAN memberships, the virtual switch's Port VLAN ID (pvid) becomes the default VLAN ID for the ports connection. This identifier/tag is added for incoming untagged messages at the port (ingress tagging). For outgoing messages with this identifier, the tag is removed at the port (egress untagging, depending on the Vlan Membership.sendActivity). **Tags:** atp.Splitkey=defaultVlan
macAddress VlanAssignment	MacAddressVlan Membership	*	aggr	Statically defines the assignment of MAC-Multicast-Addresses, optionally together with VLANs, to this CouplingPort. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=macAddressVlanAssignment, macAddress VlanAssignment.shortName, macAddressVlan Assignment.variationPoint.shortLabel vh.latestBindingTime=postBuild
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the CouplingPort.
macSecProps	MacSecProps	*	aggr	Properties to configure MACsec (Media access control security) and the MKA (MACsec Key Agreement) for the CouplingPort (PHY). **Tags:** atp.Splitkey=macSecProps atp.Status=candidate
physicalLayer Type	EthernetPhysicalLayer TypeEnum	0..1	attr	Specifies the physical layer type of the CouplingPort.

▽

△

Class	**CouplingPort**			
plcaProps	PlcaProps	0..1	aggr	Optional properties for configuration of PLCA (Physical Layer Collision Avoidance) in case 10-BASE-T1S Ethernet is used and PLCA is enabled on the Coupling Port (PHY). **Tags:** atp.Splitkey=plcaProps
pncMapping	PncMappingIdent	*	ref	Reference to the partial networks this CouplingPort participates in. **Tags:** atp.Splitkey=pncMapping
receiveActivity	EthernetSwitchVlan IngressTagEnum	0..1	attr	Defines the handling of frames at the ingress port.
vlan Membership	VlanMembership	*	aggr	Messages of VLANs that are defined here can be communicated via the CouplingPort. **Tags:** atp.Splitkey=vlanMembership
wakeupSleep OnDataline Config	EthernetWakeupSleep OnDatalineConfig	0..1	ref	Optional reference to EthernetWakeupSleepOnDataline Config. **Tags:** atp.Splitkey=wakeupSleepOnDatalineConfig

Table A.1: CouplingPort

Class	**Dhcpv6Props**			
Note	This meta-class specifies the configuration options for DHCPv6.			
Base	*ARObject*			
Aggregated by	Ipv6Props.dhcpProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpDhcpV6 CnfDelayMax	TimeValue	0..1	attr	Maximum delay in seconds before sending the first Confirm message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval.
tcpIpDhcpV6 CnfDelayMin	TimeValue	0..1	attr	Minimum delay in seconds before the first Confirm message will be sent.
tcpIpDhcpV6Inf DelayMax	TimeValue	0..1	attr	Maximum delay in seconds before sending the first Information Request message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval.
tcpIpDhcpV6Inf DelayMin	TimeValue	0..1	attr	Minimum delay (s) before the first Information Request message will be sent.
tcpIpDhcpV6Sol DelayMax	TimeValue	0..1	attr	Maximum delay in seconds before sending the first Solicit message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval.
tcpIpDhcpV6Sol DelayMin	TimeValue	0..1	attr	Minimum delay (s) before the first Solicit message will be sent.

Table A.2: Dhcpv6Props

Class	**EthTcpIpIcmpProps**
Note	This meta-class is used to configure the Machine specific ICMP (Internet Control Message Protocol) attributes **Tags:** atp.recommendedPackage=EthTcpIcmpProps
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*

▽

△

Class	**EthTcpIpIcmpProps**			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
icmpV4Props	TcpIpIcmpv4Props	0..1	aggr	ICMPv4 configuration properties **Tags:** atp.Splitkey=icmpV4Props
icmpV6Props	TcpIpIcmpv6Props	0..1	aggr	ICMPv6 configuration properties **Tags:** atp.Splitkey=icmpV6Props

Table A.3: EthTcpIpIcmpProps

Class	**EthTcpIpProps**			
Note	This meta-class is used to configure the Machine specific TcpIp Stack attributes. **Tags:** atp.recommendedPackage=EthTcpIpProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpProps	TcpProps	0..1	aggr	TCP configuration properties **Tags:** atp.Splitkey=tcpProps
udpProps	UdpProps	0..1	aggr	UDP configuration properties **Tags:** atp.Splitkey=udpProps

Table A.4: EthTcpIpProps

Class	«atpVariation» **EthernetCommunicationController**			
Note	Ethernet specific communication port attributes.			
Base	*ARObject*, *CommunicationController*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.commController, MachineDesign.communicationController			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
canXlConfig	AbstractCan Communication Controller	0..1	ref	If the Ethernet frames handled by this Ethernet CommunicationController are to be tunneled through CAN XL, then this reference shall refer to the Abstract CanCommunicationController that aggregates the Can ControllerXlConfiguration of the physical CAN XL channel to be used for tunneling. **Tags:** atp.Splitkey=canXlConfig
couplingPort	CouplingPort	*	aggr	Optional CouplingPort that can be used to connect the ECU to a CouplingElement (e.g. a switch). **Tags:** atp.Splitkey=couplingPort, couplingPort.shortName
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the ethernet controller.
macUnicast Address	MacAddressString	0..1	attr	Media Access Control address (MAC address) that uniquely identifies each EthernetCommunication Controller in the network.
maximum ReceiveBuffer Length	Integer	0..1	attr	Determines the maximum receive buffer length (frame length) in bytes.

▽

△

Class	«atpVariation» **EthernetCommunicationController**			
maximum TransmitBuffer Length	Integer	0..1	attr	Determines the maximum transmit buffer length (frame length) in bytes.
slaveActAs Passive Communication Slave	Boolean	0..1	attr	This attribute specifies if the EcuInstance is acting as a passive communication slave on the connected Physical Channel. This is used for EthernetCommunication Controllers that use Ethernet hardware which supports wake-up and sleep on the network (e.g. Open Alliance TC10 compliant Ethernet hardware).
slaveQualified UnexpectedLink DownTime	TimeValue	0..1	attr	This attribute specifies time when an unexpected link down is evaluated as link down and indicated to the AUTOSAR communication stack.

Table A.5: EthernetCommunicationController

Class	**Ipv4ArpProps**			
Note	Specifies the configuration options for the ARP (Address Resolution Protocol).			
Base	*ARObject*			
Aggregated by	Ipv4Props.arpProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpArpNum GratuitousArp OnStartup	PositiveInteger	0..1	attr	This attribute specifies the number of gratuitous ARP replies which shall be sent on assignment of a new IP address.
tcpIpArpPacket QueueEnabled	Boolean	0..1	attr	This attribute enables (TRUE) or disables (FALSE) support of the ARP Packet Queue according to IETF RFC 1122, section 2.3.2.2.
tcpIpArp Request Timeout	TimeValue	0..1	attr	This attribute specifies a timeout in seconds for the validity of ARP requests. After the transmission of an ARP request the TcpIp shall skip the transmission of any further ARP requests to the same destination within a duration of tcpIpArpRequestTimeout seconds. (IETF RFC 1122, section 2.3.2.1).
tcpIpArpTable EntryTimeout	TimeValue	0..1	attr	This attribute specifies the timeout in seconds after which an unused ARP entry is removed.

Table A.6: Ipv4ArpProps

Class	**Ipv4AutoIpProps**			
Note	Specifies the configuration options for Auto-IP (automatic private IP addressing).			
Base	*ARObject*			
Aggregated by	Ipv4Props.autoIpProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpAutoIpInit Timeout	TimeValue	0..1	attr	This attribute specifies the time in seconds Auto-IP waits at startup, before beginning with ARP probing. This delay is used to give DHCP time to acquire a lease in case a DHCP server is present.

Table A.7: Ipv4AutoIpProps

Class	**Ipv4FragmentationProps**			
Note	Specifies the configuration options for IPv4 packet fragmentation/reassembly.			
Base	*ARObject*			
Aggregated by	Ipv4Props.fragmentationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpIp Fragmentation RxEnabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support for reassembling of incoming datagrams that are fragmented according to IETF RFC 815 (IP Datagram Reassembly Algorithms).
tcpIpIpNum Fragments	PositiveInteger	0..1	attr	Specifies the maximum number of IP fragments per datagram.
tcpIpIpNum ReassDgrams	PositiveInteger	0..1	attr	Specifies the maximum number of fragmented IP datagrams that can be reassembled in parallel.
tcpIpIpReass Timeout	TimeValue	0..1	attr	Specifies the timeout in [s] after which an incomplete datagram gets discarded.

Table A.8: Ipv4FragmentationProps

Class	**Ipv4Props**			
Note	This meta-class specifies the configuration options for IPv4.			
Base	*ARObject*			
Aggregated by	EthIpProps.ipv4Props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arpProps	Ipv4ArpProps	0..1	aggr	Configuration properties for the ARP (Address Resolution Protocol). **Tags:** atp.Splitkey=arpProps
autoIpProps	Ipv4AutoIpProps	0..1	aggr	Configuration options for Auto-IP (automatic private IP addressing). **Tags:** atp.Splitkey=autoIpProps
fragmentation Props	Ipv4Fragmentation Props	0..1	aggr	Configuration options for IPv4 packet fragmentation/ reassembly. **Tags:** atp.Splitkey=fragmentationProps

Table A.9: Ipv4Props

Class	**Ipv6NdpProps**			
Note	This meta-class specifies the configuration options for the Neighbor Discovery Protocol for IPv6.			
Base	*ARObject*			
Aggregated by	Ipv6Props.ndpProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpNdpDefault ReachableTime	TimeValue	0..1	attr	Configuration of the ReachableTime (s) specified in [RFC4861 6.3.2. Host Variables].
tcpIpNdpDefault RetransTimer	TimeValue	0..1	attr	Configures the default value (s) for the RetransTimer variable specified in [RFC4861 6.3.2. Host Variables].
tcpIpNdpDefault RouterListSize	PositiveInteger	0..1	attr	Maximum number of default router entries.

▽

△

Class	**Ipv6NdpProps**			
tcpIpNdp Defensive Processing	Boolean	0..1	attr	If enabled the NDP shall only process Neighbor Advertisements which are received in reaction to a previously transmitted Neighbor Solicitation as well as skipping updates to the Neighbor Cache based on received Neighbor Solicitations. If disabled all Neighbor Advertisements and Solicitations shall be processed as specified in RFC4861.
tcpIpNdpDelay FirstProbeTime Value	TimeValue	0..1	attr	Delay before sending the first NUD probe in (s).
tcpIpNdp Destination CacheSize	PositiveInteger	0..1	attr	Maximum number of entries in the destination cache.
tcpIpNdp DynamicHop LimitEnabled	Boolean	0..1	attr	If enabled the default hop limit may be reconfigured based on received Router Advertisements.
tcpIpNdp DynamicMtu Enabled	Boolean	0..1	attr	Allow dynamic reconfiguration of link MTU via Router Advertisements.
tcpIpNdp Dynamic ReachableTime Enabled	Boolean	0..1	attr	If enabled the default Reachable Time value may be reconfigured based on received Router Advertisements.
tcpIpNdp Dynamic RetransTime Enabled	Boolean	0..1	attr	If enabled the default Retransmit Timer value may be reconfigured based on received Router Advertisements.
tcpIpNdpMax RandomFactor	PositiveInteger	0..1	attr	Maximum random factor used for randomization
tcpIpNdpMaxRtr Solicitation Delay	TimeValue	0..1	attr	Maximum delay before the first Router Solicitation will be sent after interface initialization in (s).
tcpIpNdpMaxRtr Solicitations	PositiveInteger	0..1	attr	Maximum number of Router Solicitations that will be sent before the first Router Advertisement has been received.
tcpIpNdpMin RandomFactor	PositiveInteger	0..1	attr	Minimum random factor used for randomization
tcpIpNdp Neighbor Unreachability Detection Enabled	Boolean	0..1	attr	Neighbor Unreachability Detection is used to remove unused entries from the neighbor cache. This feature is a basic feature of NDP and should be turned on.
tcpIpNdpNum Multicast Solicitations	PositiveInteger	0..1	attr	Maximum number of multicast solicitations that will be sent when performing address resolution.
tcpIpNdpNum Unicast Solicitations	PositiveInteger	0..1	attr	Maximum number of unicast solicitations that will be sent when performig Neighbor Unreachability Detection.
tcpIpNdpPacket QueueEnabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of a NDP Packet Queue according to IETF RFC 4861, section 7.2.2.
tcpIpNdpPrefix ListSize	PositiveInteger	0..1	attr	Maximum number of entries in the on-link prefix list.
tcpIpNdp Random ReachableTime Enabled	Boolean	0..1	attr	If enabled the value of ReachableTime will be multiplied with a random value between MIN_RANDOM_FACTOR and MAX_RANDOM_FACTOR in order to prevent multiple nodes from transmitting at exactly the same time.

▽

△

Class	**Ipv6NdpProps**			
tcpIpNdpRndRtr Solicitation DelayEnabled	Boolean	0..1	attr	If enabled the first router solicitation will be delayed randomly from [0...MAX_RTR_SOLICITATION_DELAY]. Otherwise the first router solicitation will be sent after exactly MAX_RTR_SOLICITATION_DELAY milliseconds.
tcpIpNdpRtr Solicitation Interval	TimeValue	0..1	attr	Interval between consecutive Router Solicitations in (s).
tcpIpNdpSlaac DadNumberOf Transmissions	PositiveInteger	0..1	attr	Number of Neighbor Solicitations that have to be unanswered in order to set an autoconfigurated address to PREFERRED (usable) state.
tcpIpNdpSlaac Dad Retransmission Delay	TimeValue	0..1	attr	Sets the maximum value for the address configuration delay (s).
tcpIpNdpSlaac DelayEnabled	Boolean	0..1	attr	If enabled transmission of the first DAD Neighbor Solicitation will be delayed by a random value from [0...MAX_DAD_DELAY].
tcpIpNdpSlaac OptimisticDad Enabled	Boolean	0..1	attr	Enable Optimistic Duplicate Address Detection (DAD) according to RFC4429.

Table A.10: Ipv6NdpProps

Class	**Ipv6Props**			
Note	This meta-class specifies the configuration options for IPv6.			
Base	*ARObject*			
Aggregated by	EthIpProps.ipv6Props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dhcpProps	Dhcpv6Props	0..1	aggr	Configuration properties for DHCPv6. **Tags:** atp.Splitkey=dhcpProps
fragmentation Props	Ipv6Fragmentation Props	0..1	aggr	Configuration properties for IPv6 packet fragmentation/ reassembly. **Tags:** atp.Splitkey=fragmentationProps
ndpProps	Ipv6NdpProps	0..1	aggr	Configuration properties for the Neighbor Discovery Protocol for IPv6. **Tags:** atp.Splitkey=ndpProps

Table A.11: Ipv6Props

Class	**MacSecLocalKayProps**			
Note	Configuration of the MAC Security Key Agreement Entity (KaY). **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	MacSecProps.macSecKayConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationMac Address	MacAddressString	0..1	attr	This attribute defines the destination MAC Address that is used to calculate the ICV (Integrity Check Value). **Tags:** atp.Status=candidate

▽

△

Class	**MacSecLocalKayProps**			
globalKayProps	MacSecGlobalKay Props	0..1	ref	Reference to properties that are shared between MAC Security Key Agreement Entities. **Tags:** atp.Splitkey=globalKayProps atp.Status=candidate
keyServer Priority	PositiveInteger	0..1	attr	This attribute defines the key-server priority. **Tags:** atp.Status=candidate
mkaParticipant	MacSecKayParticipant	*	ref	Reference to MKA participant settings supported on the CouplingPort. **Tags:** atp.Splitkey=mkaParticipant atp.Status=candidate
role	MacSecRoleEnum	0..1	attr	Role of the MAC Security Key Agreement Entity **Tags:** atp.Status=candidate
sourceMac Address	MacAddressString	0..1	attr	This attribute defines the source MAC Address that is used to calculate the ICV (Integrity Check Value). **Tags:** atp.Status=candidate

Table A.12: MacSecLocalKayProps

Class	**MacSecProps**			
Note	This meta-class allows to configure MACsec (Media access control security) and the MKA (MACsec Key Agreement) for the CouplingPort (PHY). **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	CouplingPort.macSecProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
autoStart	Boolean	0..1	attr	This attribute defines how the Port Access Entity (PAE) is started: • true := Autostart • false := Manual Start **Tags:** atp.Status=candidate
macSecKay Config	MacSecLocalKayProps	0..1	aggr	Properties to configure the MKA instance (KaY) for a controlled CouplingPort (PaE). **Tags:** atp.Splitkey=macSecKayConfig atp.Status=candidate
onFail Permissive Mode	MacSecFailPermissive ModeEnum	0..1	attr	This attribute sets the behavior of the Port Access Entity in case MACsec does not succeed. **Tags:** atp.Status=candidate
onFail Permissive ModeTimeout	TimeValue	0..1	attr	Timeout in seconds to enable the controlled port in case onFailPermissiveMode is set to Timeout. **Tags:** atp.Status=candidate
sakRekeyTime Span	TimeValue	0..1	attr	Time in seconds to trigger the rekey of an in use SAK (Static Secure Association key). If set to 0, the rekey will not be triggered after a time span. **Tags:** atp.Status=candidate

Table A.13: MacSecProps

Class	**Machine**			
Note	Machine that represents an Adaptive Autosar Software Stack. **Tags:** atp.recommendedPackage=Machines This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
default Application Timeout	EnterExitTimeout	0..1	aggr	This aggregation defines a default timeout in the context of a given Machine with respect to the launching and termination of applications. **Tags:** atp.Splitkey=defaultApplicationTimeout
environment Variable	TagWithOptionalValue	*	aggr	This aggregation represents the collection of environment variables that shall be added to the environment defined on the level of the enclosing Machine. **Tags:** atp.Splitkey=environmentVariable
machineDesign	MachineDesign	0..1	ref	Reference to the MachineDesign this Machine is implementing. **Tags:** atp.Splitkey=machineDesign
module Instantiation	AdaptiveModule Instantiation	*	aggr	Configuration of Adaptive Autosar module instances that are running on the machine. **Tags:** atp.Splitkey=moduleInstantiation, module Instantiation.shortName
processor	Processor	*	aggr	This represents the collection of processors owned by the enclosing machine. **Tags:** atp.Splitkey=processor, processor.shortName
secure Communication Deployment	SecureCommunication Deployment	*	aggr	Target-configuration of secure communication protocol configuration settings to crypto module entities. **Tags:** atp.Splitkey=secureCommunicationDeployment, secure CommunicationDeployment.shortName atp.Status=obsolete
trustedPlatform Executable LaunchBehavior	TrustedPlatform ExecutableLaunch BehaviorEnum	0..1	attr	This attribute controls the behavior of how authentication affects the ability to launch for each Executable.

Table A.14: Machine

Class	**MachineDesign**			
Note	This meta-class represents the ability to define requirements on a Machine in the context of designing a system. **Tags:** atp.recommendedPackage=MachineDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Fibex Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesign Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accessControl	AccessControlEnum	0..1	attr	This attribute defines how the access restriction to the Service Instance is defined.



△

Class	**MachineDesign**			
clientIdRange	ClientIdRange	0..1	aggr	Optional restriction of Client Identifiers for this Machine to an allowed range of numerical values. **Tags:** atp.Splitkey=clientIdRange
communication Connector	Communication Connector	*	aggr	This aggregation defines the network connection of the machine. **Tags:** atp.Splitkey=communicationConnector, communicationConnector.shortName
communication Controller	Communication Controller	*	aggr	CommunicationControllers of the Machine that are used for description of 10-Base-T1S topologies **Tags:** atp.Splitkey=communicationController, communicationController.shortName
cryptoKeySlot	CryptoKeySlotDesign	*	aggr	This aggregation represents the key slots for which a key slot design is created in the context of the enclosing machine design. **Tags:** atp.Splitkey=cryptoKeySlot, cryptoKeySlot.short Name
ethIpProps	EthIpProps	*	ref	Machine specific IP attributes. **Tags:** atp.Splitkey=ethIpProps
functional ClusterDesign	AbstractFunctional ClusterDesign	*	aggr	Configuration settings for Functional Clusters on the machine design level. **Tags:** atp.Splitkey=functionalClusterDesign, functional ClusterDesign.shortName
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pnResetTimer	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests.
processor	Processor	*	aggr	This represents the collection of processors owned by the enclosing MachineDesign. **Tags:** atp.Splitkey=processor, processor.shortName
service Discovery Config	ServiceDiscovery Configuration	*	aggr	Set of service discovery configuration settings that are defined on the machine for individual Communication Connectors. **Tags:** atp.Splitkey=serviceDiscoveryConfig
tcpIpIcmpProps	EthTcpIpIcmpProps	*	ref	Machine specific ICMP (Internet Control Message Protocol) attributes **Tags:** atp.Splitkey=tcpIpIcmpProps
tcpIpProps	EthTcpIpProps	*	ref	Machine specific TcpIp Stack attributes. **Tags:** atp.Splitkey=tcpIpProps

Table A.15: MachineDesign

Class	**TcpIpIcmpv4Props**			
Note	This meta-class specifies the configuration options for ICMPv4 (Internet Control Message Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpIcmpProps.icmpV4Props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpIcmpV4 EchoReply Enabled	Boolean	0..1	attr	This attribute enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception.

▽

△

Class	TcpIpIcmpv4Props			
tcpIpIcmpV4Ttl	PositiveInteger	0..1	attr	This attribute is only relevant in case that ICMP (Internet Control Message Protocol) is used. It specifies the default Time-to-live value of outgoing ICMP packets.

Table A.16: TcpIpIcmpv4Props

Class	TcpIpIcmpv6Props			
Note	This meta-class specifies the configuration options for ICMPv6 (Internet Control Message Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpIcmpProps.icmpV6Props			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpIpIcmpV6 EchoReply Avoid Fragmentation	Boolean	0..1	attr	This attribute defines whether the echo reply is only transmitted in case that the incoming ICMPv6 Echo Request (Pings) fits the MTU of the respective interface, i.e. can be transmitted without IPv6 fragmentation.
tcpIpIcmpV6 EchoReply Enabled	Boolean	0..1	attr	This attribute enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception.
tcpIpIcmpV6 HopLimit	PositiveInteger	0..1	attr	Default Hop-Limit value of outgoing ICMPv6 packets.
tcpIpIcmpV6 MsgDestination Unreachable Enabled	Boolean	0..1	attr	This attribute Enables/Disables the transmission of Destination Unreachable Messages.
tcpIpIcmpV6 MsgParameter Problem Enabled	Boolean	0..1	attr	If enabled an ICMPv6 parameter problem message will be sent if a received packet has been dropped due to unknown options or headers that are found in the packet.

Table A.17: TcpIpIcmpv6Props

Class	TcpProps			
Note	This meta-class specifies the configuration options for TCP (Transmission Control Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpProps.tcpProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpCongestion Avoidance Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP congestion avoidance algorithm according to IETF RFC 5681.
tcpDelayedAck Timeout	TimeValue	0..1	attr	The maximal time an acknowledgement is delayed for transmission in seconds.
tcpFast Recovery Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP Fast Recovery according to IETF RFC 5681.
tcpFast Retransmit Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP Fast Retransmission according to IETF RFC 5681.
tcpFinWait2 Timeout	TimeValue	0..1	attr	Timeout in [s] to receive a FIN from the remote node (after this node has initiated connection termination), i.e. maximum time waiting in FINWAIT-2 for a connection termination request from the remote TCP.
tcpKeepAlive Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) TCP Keep Alive Probes according to IETF RFC 1122 chapter 4.2.3.6.

▽

△

Class	**TcpProps**			
tcpKeepAlive Interval	TimeValue	0..1	attr	Specifies the interval in seconds between subsequent keepalive probes.
tcpKeepAlive ProbesMax	PositiveInteger	0..1	attr	Maximum number of times that a TCP Keep Alive is retransmitted before the connection is closed.
tcpKeepAlive Time	TimeValue	0..1	attr	Specifies the time in [s] between the last data packet sent (simple ACKs are not considered data) and the first keepalive probe.
tcpMaxRtx	PositiveInteger	0..1	attr	Maximum number of times that a TCP segment is retransmitted before the TCP connection is closed. This parameter is only valid if tcpRetransmissionTimeout is configured. Note: This parameter also applies for FIN retransmissions.
tcpMsl	TimeValue	0..1	attr	Maximum segment lifetime in [s].
tcpNagle Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of Nagle's algorithm according to IETF RFC 1122 (chapter 4.2.3.4 When to Send Data). If enabled the Nagle's algorithm is activated per default for all TCP sockets, but can be deactivated per Socket (with the attribute TcpTp.nagle Algorithm).
tcpReceive WindowMax	PositiveInteger	0..1	attr	Default value of maximum receive window in bytes.
tcp Retransmission Timeout	TimeValue	0..1	attr	Timeout in [s] before an unacknowledged TCP segment is sent again. If the timeout is disabled, no TCP segments shall be retransmitted.
tcpSlowStart Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP slow start algorithm according to IETF RFC 5681.
tcpSynMaxRtx	PositiveInteger	0..1	attr	Maximum number of times that a TCP SYN is retransmitted.
tcpSynReceived Timeout	TimeValue	0..1	attr	Timeout in [s] to complete a remotely initiated TCP connection establishment, i.e. maximum time waiting in SYN-RECEIVED for a confirming connection request acknowledgement after having both received and sent a connection request.
tcpTtl	PositiveInteger	0..1	attr	Default Time-to-live value of outgoing TCP packets.

Table A.18: TcpProps

Enumeration	**TrustedPlatformExecutableLaunchBehaviorEnum**
Note	This enumeration provides options for controlling the behavior of how authentication affects the ability to launch an Executable. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	Machine.trustedPlatformExecutableLaunchBehavior
Literal	***Description***
monitorMode	An Executable shall always launch, even if the corresponding authentication fails **Tags:** atp.EnumerationLiteralIndex=1
noTrustedPlatform Support	This value shall be used if there is no TrustedPlatform support on the Machine **Tags:** atp.EnumerationLiteralIndex=2
strictMode	An Executable shall not launch if the corresponding authentication fails. **Tags:** atp.EnumerationLiteralIndex=0

Table A.19: TrustedPlatformExecutableLaunchBehaviorEnum

Class	**UdpProps**			
Note	This meta-class specifies the configuration options for UDP (User Datagram Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpProps.udpProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
udpTtl	PositiveInteger	0..1	attr	Default Time-to-live value of outgoing UDP packets.

Table A.20: UdpProps

B Demands and constraints on Base Software (normative)

This specification does not specify a functional cluster. This subchapter does not apply.

C Platform Extension Interfaces (normative)

This specification does not specify a functional cluster. This subchapter does not apply.

D Not implemented requirements

This functional cluster implements all functional requirements specified in the corresponding requirement specifications.

E History of Constraints and Specification Items

This chapter provides an overview of the history of constraints and specification items. Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

E.1 Change History of this document according to AUTOSAR Release R26-11

E.1.1 Added Specification Items in R26-11

Number	Heading
[SWS_APBSW_00001]	Security Events for Base Software
[SWS_APBSW_00002]	Security events for ap_core
[SWS_APBSW_00070]	Reporting Security Event with Verification Mode set to MEASURED_BOOT to IdsM
[SWS_APBSW_00071]	Reporting Security Event with Verification Mode set to STRICT to IdsM
[SWS_APBSW_00072]	Security event context data definition: SEV_EXEC_SW_COMPONENT_INTEGRITY_CHECK_FAILED
[SWS_APBSW_00100]	Security event context data definition: SEV_ETH_DROP_UNKNOWN_ETHERTYPE
[SWS_APBSW_00101]	Security event context data definition: SEV_ETH_DROP_VLAN_DOUBLE_TAG
[SWS_APBSW_00102]	Security event context data definition: SEV_ETH_DROP_INV_VLAN
[SWS_APBSW_00103]	Security event context data definition: SEV_ETH_DROP_MAC_COLLISION
[SWS_APBSW_00300]	Security event context data definition: SEV_ARP_IP_ADDR_CONFLICT
[SWS_APBSW_00301]	Security event context data definition: SEV_TCP_DROP_INV_DEST_PORT
[SWS_APBSW_00302]	Security event context data definition: SEV_UDP_DROP_INV_DEST_PORT
[SWS_APBSW_00303]	Security event context data definition: SEV_IPV4_DROP_INV_DEST_ADDR
[SWS_APBSW_00304]	Security event context data definition: SEV_IPV6_DROP_INV_DEST_ADDR



△

Number	Heading
[SWS_APBSW_00500]	Security event context data definition: SEV_MKA_AUTHENTICATION_FAILURE
[SWS_APBSW_00501]	Security event context data definition: SEV_MKA_TIMEOUT
[SWS_APBSW_00502]	Security event context data definition: SEV_MKA_PORT_NOT_ENABLED
[SWS_APBSW_00503]	Security event context data definition: SEV_MKA_CIPHER_SUITE_NOT_SUPPORTED
[SWS_APBSW_00504]	Security event context data definition: SEV_MKA_PORT_NUMBER_CHANGE
[SWS_APBSW_02000]	Log OS tracing started
[SWS_APBSW_02001]	Log Task Schedule Notification
[SWS_APBSW_02002]	Log Task Wait Notification
[SWS_APBSW_02003]	Log Task Release Notification
[SWS_APBSW_02004]	Log Task Preempt Notification
[SWS_APBSW_02005]	Log Task Exit Notification
[SWS_APBSW_02006]	Log Task Creation Notification
[SWS_APBSW_02007]	Log Task Renaming Notification
[SWS_APBSW_02030]	Log Message OsTaskCreate
[SWS_APBSW_02031]	Log Message OsTaskTerminate
[SWS_APBSW_02032]	Log Message OsTaskInfo
[SWS_APBSW_02033]	Log Message OsTaskPreempt
[SWS_APBSW_02034]	Log Message OsTaskRelease
[SWS_APBSW_02035]	Log Message OsTaskRename
[SWS_APBSW_02036]	Log Message OsTaskSchedule
[SWS_APBSW_02037]	Log Message OsTaskWait

▽

△

Number	Heading
[SWS_APBSW_02040]	Log Process Switch Notification
[SWS_APBSW_02041]	Log Process Creation Notification
[SWS_APBSW_02042]	Log Process Ending Notification
[SWS_APBSW_02043]	Log Process Renaming Notification
[SWS_APBSW_02050]	Log Message OsProcessCreate
[SWS_APBSW_02051]	Log Message OsProcessDestroy
[SWS_APBSW_02052]	Log Message OsProcessInfo
[SWS_APBSW_02053]	Log Message OsProcessRename
[SWS_APBSW_02054]	Log Message OsProcessSwitch
[SWS_APBSW_03000]	Base SW MACsec support
[SWS_APBSW_03001]	MACsec configuration
[SWS_APBSW_03002]	Daemon start-up
[SWS_APBSW_03003]	Link indication
[SWS_APBSW_03500]	Availability of a Trust Anchor
[SWS_APBSW_03501]	Integrity and Authenticity of Machine configuration
[SWS_APBSW_03502]	Integrity and Authenticity of each `Executable`
[SWS_APBSW_03503]	Integrity and Authenticity of shared objects
[SWS_APBSW_03504]	Integrity and Authenticity of Processed Manifest configurations
[SWS_APBSW_03505]	Failed authenticity checks
[SWS_APBSW_03506]	Launch Behavior Validation
[SWS_APBSW_03507]	Monitor Mode

▽

△

Number	Heading
[SWS_APBSW_03508]	Strict Mode - Execution manifest
[SWS_APBSW_03509]	Strict Mode - Service Instance manifests
[SWS_APBSW_03510]	Strict Mode - Executables

Table E.1: Added Specification Items in R26-11

E.1.2 Changed Specification Items in R26-11

none

E.1.3 Deleted Specification Items in R26-11

none