Document Title	Specification of Firewall for Adaptive Platform
Document Category	Software specification (SWS)
Document UID	1063
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	• Removed upwards tracing to `FO_RS_Firewall` • Restructured related documentation
2025-11-27	R25-11	AUTOSAR Release Management	• Updated context data for security events generated by the firewall • PortInterface to API class binding introduced • Added access control for FirewallStateSwitchInterface and corresponding security event for access violations • Fixed various issues (missing thread-safety definition, obsolete tracing)
2024-11-27	R24-11	AUTOSAR Release Management	• Various API changes (specification of error domain, thread-safety, error recoverability, ...) • Migration to new SW template • Updated SEv context data specification table
2023-11-23	R23-11	AUTOSAR Release Management	• Minor bugfixes
2022-11-24	R22-11	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 6

2 Acronyms and Abbreviations 7
2.1 Acronyms . . . 7
2.2 Abbreviations . . . 7

3 Related documentation 8
3.1 Input documents & related standards and norms . . . 8
3.2 Further applicable specification . . . 8

4 Constraints and assumptions 9
4.1 Known limitations . . . 9

5 Dependencies to other Functional Clusters 10
5.1 Provided Interfaces . . . 10
5.2 Required Interfaces . . . 11

6 Requirements Tracing 12

7 Functional specification 13
7.1 Architecture Overview . . . 13
7.2 Network packet inspection . . . 15
7.2.1 Stateless packet inspection . . . 16
7.2.1.1 Inspection of not modeled protocols . . . 17
7.2.2 Stateful packet inspection . . . 18
7.2.3 Deep packet inspection . . . 18
7.2.3.1 SOME/IP . . . 18
7.2.3.2 DDS . . . 20
7.2.3.3 DoIP . . . 20
7.2.3.4 Generic inspection . . . 21
7.3 Network packet filtering . . . 21
7.3.1 Allowlists and Blocklists . . . 21
7.3.2 Rate limiting . . . 22
7.3.3 State dependent filtering . . . 23
7.4 Firewall Rule Management . . . 24
7.5 Functional cluster life-cycle . . . 25
7.5.1 Startup . . . 25
7.5.2 Shutdown . . . 25
7.5.3 Daemon crash . . . 25
7.6 Reporting . . . 25
7.6.1 Security Events . . . 25
7.6.1.1 SEvs raised by the firewall . . . 26
7.6.1.2 Raising SEvs . . . 33
7.6.2 Log Messages . . . 36

7.6.3 Violation Messages . . . 36
7.6.4 Production Errors . . . 36
8 API specification 37
8.1 PortInterface to API class binding . . . 37
8.2 API Header Files . . . 38
8.3 API Common Data Types . . . 38
8.4 API Reference . . . 39
8.4.1 FirewallStateSwitchInterface . . . 39
8.4.2 FirewallErrorDomain . . . 42
8.4.2.1 ara::fw::FwErrc . . . 42
8.4.2.2 ara::fw::GetFwErrorDomain . . . 42
8.4.2.3 ara::fw::MakeErrorCode overload for ara::fw::GetFwErrorDomain 43
8.4.2.4 ara::fw::FwException . . . 43
8.4.2.5 ara::fw::FwErrorDomain . . . 44
9 Service Interfaces 47
10 Configuration 48
10.1 Default Values . . . 48
10.2 Semantic Constraints . . . 48
A Mentioned Manifest Elements 49
B Demands and constraints on Base Software (normative) 59
C Platform Extension Interfaces (normative) 60
D Not implemented requirements 61
E Change history of AUTOSAR traceable items 62
E.1 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 62
E.1.1 Added Specification Items in R26-11 . . . 62
E.1.2 Changed Specification Items in R26-11 . . . 62
E.1.3 Deleted Specification Items in R26-11 . . . 62
E.1.4 Added Constraints in R26-11 . . . 62
E.1.5 Changed Constraints in R26-11 . . . 62
E.1.6 Deleted Constraints in R26-11 . . . 62
E.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 63
E.2.1 Added Specification Items in R25-11 . . . 63
E.2.2 Changed Specification Items in R25-11 . . . 63
E.2.3 Deleted Specification Items in R25-11 . . . 63
E.2.4 Added Constraints in R25-11 . . . 63
E.2.5 Changed Constraints in R25-11 . . . 63
E.2.6 Deleted Constraints in R25-11 . . . 63

E.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 64
E.3.1 Added Specification Items in R24-11 . . . 64
E.3.2 Changed Specification Items in R24-11 . . . 64
E.3.3 Deleted Specification Items in R24-11 . . . 64
E.3.4 Added Constraints in R24-11 . . . 64
E.3.5 Changed Constraints in R24-11 . . . 64
E.3.6 Deleted Constraints in R24-11 . . . 64
E.4 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 65
E.4.1 Added Specification Items in R23-11 . . . 65
E.4.2 Changed Specification Items in R23-11 . . . 65
E.4.3 Deleted Specification Items in R23-11 . . . 65
E.5 Traceable item history of this document according to AUTOSAR Release R22-11 . . . 65
E.5.1 Added Specification Items in R22-11 . . . 65
E.5.2 Changed Specification Items in R22-11 . . . 66
E.5.3 Deleted Specification Items in R22-11 . . . 66

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the `FC Firewall`.

The `FC Firewall` manages and configures the host-based firewall on the ECU where the Adaptive Platform is deployed on. To this end, the FC Firewall configures the underlying Firewall engine according to the Firewall Rule configuration deployed with the manifests. Additionally, the `FC Firewall` offers interfaces to adapt the Firewall rule configuration during runtime, e.g. to adapt for different vehicle contexts or to support Intrusion Prevention Systems.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations that are only relevant to the `FC Firewall`. A general list of acronyms and abbreviations is available in the [1, AUTOSAR glossary].

2.1 Acronyms

Acronym:	Description:
Firewall	An automotive Ethernet firewall is a network security device that monitors incoming and outgoing network traffic and grants or rejects network access between two or more Electronic Control Units (ECU) or between network zones (e.g. vehicle domain (ADAS, infotainment, diagnostics etc), trusted/non-trusted zones).
FC Firewall	Abbreviation for the Functional Cluster Firewall.
Firewall Rule	Pattern of expected values for a network packet together with an associated action in case a network packet matches the pattern (e.g., block or allow the network packet).
Firewall State	The Firewall State reflects the current state of the vehicle (e.g. driving, in a diagnostic session, ...) and can be set by a user application. Based on the currently active Firewall State, a specific set of `Firewall Rule`s matching the current vehicle state is active.
Allowlist	Collection of Firewall Rules where the network packet is allowed in case of a pattern match.
Blocklist	Collection of Firewall Rules where the network packet is blocked in case of a pattern match.
OSI Layer	Network layer according to the ISO OSI model as specified in ISO/IEC 7498.

Table 2.1: Acronyms used in the scope of this Document

2.2 Abbreviations

Abbreviation:	Description:
DDS	Data Distribution Service
DDS-RTPS	DDS Real-Time Publish Subscribe Protocol
DoIP	Diagnostics over IP
IDS	Intrusion Detection System
IdsM	IDS Manager
IdsR	IDS Reporter
IP	Internet Protocol
SEv	Security Event
SOME/IP	Service oriented Middleware over IP
TCP	Transmission control protocol
UCM	Update & Configuration Management
UDP	User datagram protocol

Table 2.2: Abbreviations used in the scope of this Document

3 Related documentation

This document provides the software specification for the FC Firewall. The following document complement this specification:

- **TPS_ManifestSpecification** [2]: Specification of the Adaptive AUTOSAR Meta-Model, including the modeling of the FC Firewall.

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

[3] General Specification of Adaptive Platform
AUTOSAR_AP_SWS_General

[4] Explanation of Adaptive Platform Software Architecture
AUTOSAR_AP_EXP_SWArchitecture

[5] IEEE Standard for Ethernet
https://ieeexplore.ieee.org/document/7428776

[6] SOME/IP Protocol Specification
AUTOSAR_FO_PRS_SOMEIPProtocol

[7] SOME/IP Service Discovery Protocol Specification
AUTOSAR_FO_PRS_SOMEIPServiceDiscoveryProtocol

[8] DDS Interoperability Wire Protocol, Version 2.2
http://www.omg.org/spec/DDSI-RTPS/2.2

[9] ISO 13400-2:2019 – Road vehicles – Diagnostic communication over Internet Protocol (DoIP) – Part 2: Network and transport layer requirements and services (Edition 2, Release 2019-12)
https://www.iso.org/standard/74785.html

3.2 Further applicable specification

AUTOSAR provides a general specification [3] which is also applicable for this functional cluster. It shall be considered an additional and required specification for implementing this functional cluster.

4 Constraints and assumptions

4.1 Known limitations

Features not supported for this release:

- Firewall rule (de-)activation during runtime
- Support for OEM-defined SEVs

5 Dependencies to other Functional Clusters

AUTOSAR decided not to standardize interfaces which are exclusively used between Functional Clusters (on platform-level only), to allow efficient implementations, which might depend e.g. on the used Operating System.

This chapter provides an informative guideline of the interaction of `Firewall` with other Functional Clusters in the AUTOSAR Adaptive Platform. Section 5.1 "Provided Interfaces" lists the public interfaces provided by `Firewall` to other Functional Clusters. Section 5.2 "Required Interfaces" lists the public interfaces required by `Firewall`.

The goal is to provide a clear understanding of Functional Cluster boundaries and interaction, without specifying syntactical details. This ensures compatibility between documents specifying different Functional Clusters and supports parallel implementation of different Functional Clusters. Details of internal interfaces are up to the platform provider. Additional internal interfaces, parameters and return values can be added.

A detailed technical architecture documentation of the overall AUTOSAR Adaptive Platform is provided in [4].

5.1 Provided Interfaces

Interface	*Functional Cluster*	*Purpose*
No provided interfaces		

Table 5.1: Interfaces provided to other Functional Clusters

5.2 Required Interfaces

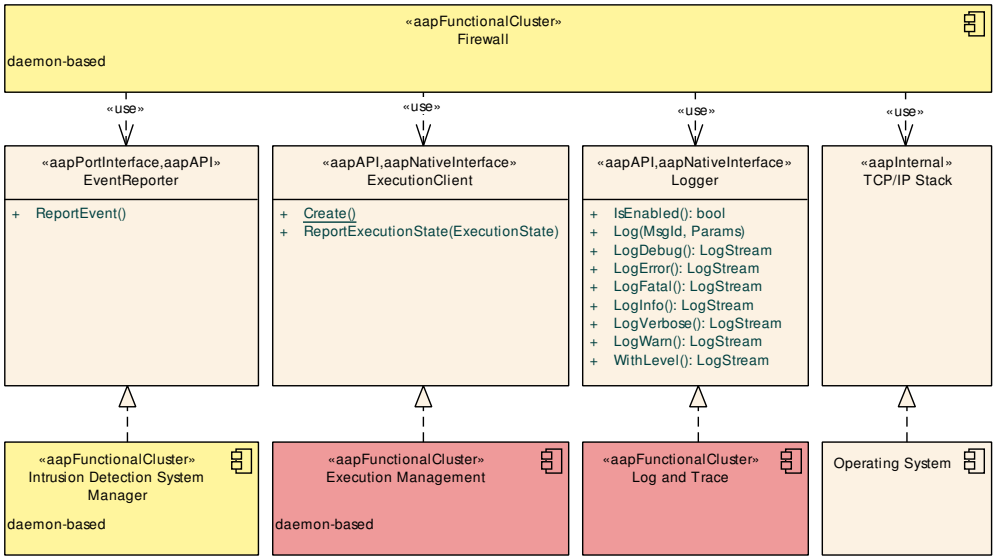


Figure 5.1: Interfaces required by Firewall from other Functional Clusters

Figure 5.1 shows the interfaces required by `Firewall` from other Functional Clusters within the AUTOSAR Adaptive Platform.

Functional Cluster	*Interface*	*Purpose*
Execution Management	ExecutionClient	The `Firewall` uses this interface to report the state of its daemon processes.
Intrusion Detection System Manager	EventReporter	The `Firewall` uses this interface to report standardized security events.
Log and Trace	Logger	This interface is used to log messages.

Table 5.2: Interfaces required from other Functional Clusters

6 Requirements Tracing

The following tables reference the requirements specified by the Adaptive Platform and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_AP_00120]	Method and Function names	[AP_SWS_Fw_83002] [AP_SWS_Fw_83003] [AP_SWS_Fw_83005] [AP_SWS_Fw_83007] [AP_SWS_Fw_83008] [AP_SWS_Fw_83009] [AP_SWS_Fw_83010] [AP_SWS_Fw_83011] [AP_SWS_Fw_83012]
[RS_AP_00121]	Parameter names	[AP_SWS_Fw_83003] [AP_SWS_Fw_83005] [AP_SWS_Fw_83011] [AP_SWS_Fw_83012]
[RS_AP_00122]	Type names	[AP_SWS_Fw_81001] [AP_SWS_Fw_83001] [AP_SWS_Fw_83004] [AP_SWS_Fw_83006]
[RS_AP_00127]	Usage of ara::core types	[AP_SWS_Fw_81001] [AP_SWS_Fw_83001] [AP_SWS_Fw_83004] [AP_SWS_Fw_83006]
[RS_AP_00130]	AUTOSAR Adaptive Platform shall represent a rich and modern programming environment	[AP_SWS_Fw_81001] [AP_SWS_Fw_83001] [AP_SWS_Fw_83002] [AP_SWS_Fw_83003] [AP_SWS_Fw_83004] [AP_SWS_Fw_83005] [AP_SWS_Fw_83006] [AP_SWS_Fw_83007] [AP_SWS_Fw_83008] [AP_SWS_Fw_83009] [AP_SWS_Fw_83010] [AP_SWS_Fw_83011] [AP_SWS_Fw_83012]
[RS_AP_00159]	usage of “noexcept” specifier	[AP_SWS_Fw_83002] [AP_SWS_Fw_83003] [AP_SWS_Fw_83005] [AP_SWS_Fw_83007] [AP_SWS_Fw_83008] [AP_SWS_Fw_83009] [AP_SWS_Fw_83010] [AP_SWS_Fw_83011]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Architecture Overview

The FC Firewall serves as a management cluster that abstracts the underlying firewall engine and configures it according to the firewall filter rules provided by the manifests. The actual filtering of the network traffic is carried out by the firewall engine, which can be realized in different ways on different levels, e.g. by inspecting traffic within the TCP/IP stack provided by the operating system, by leveraging hardware inspection capabilities and performing the inspection on hardware level or by inspecting different aspects on different layers and perform deep packet inspection at higher level closer to the application, for instance. The functional cluster firewall does not mandate a specific solution but lets the implementer choose the best solution for their use-case.

The general behaviour of a firewall can be described as follows: The FC Firewall manages a list of expected network packet patterns, where each pattern is associated with a respective action (e.g. allow or block the network packet). The combination of network packet pattern and action is called a FirewallRule. For every network packet that passes the network stack (ingress and egress), the firewall compares the network packet against the list of patterns. In case of a pattern match, the firewall carries out the action associated with the pattern. If no pattern matches (no-match case), the firewall carries out a default action.

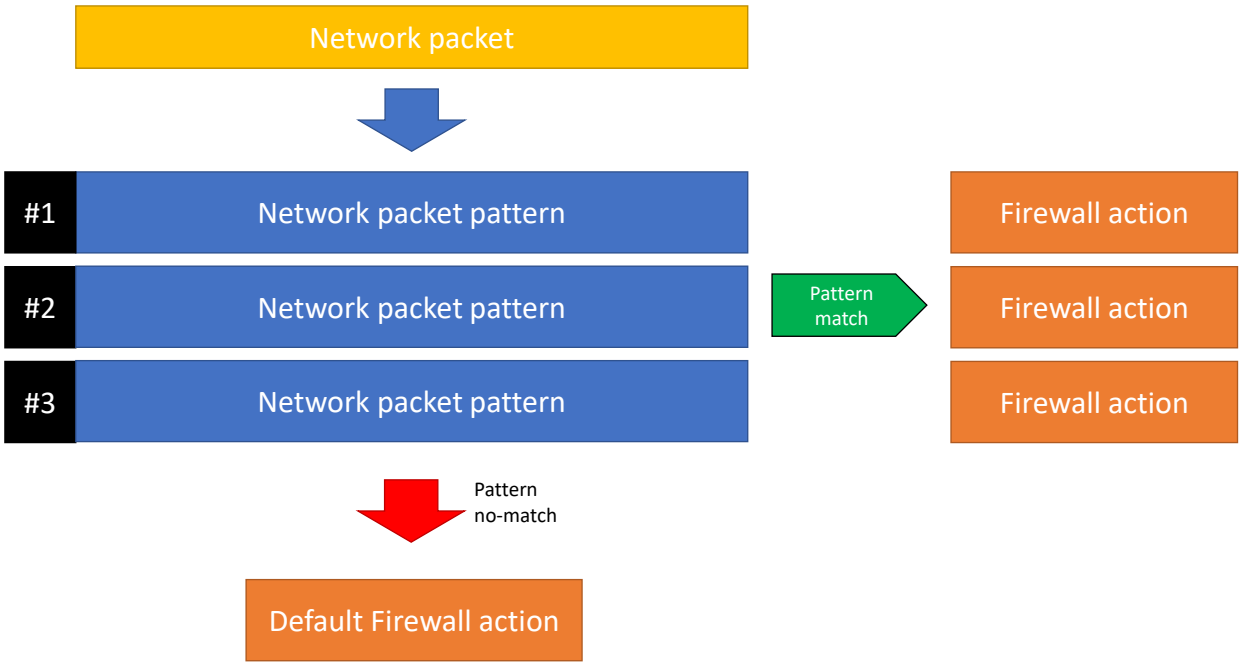


Figure 7.1: Pattern matching mechanism

The FirewallRules are deployed to the Machine via the Machine Manifest. The FC Firewall uses these FirewallRules to configure the underlying firewall en-

gine. The FirewallRules are generally static, but the FC Firewall offers a mechanism to dynamically enable/disable FirewallRules during runtime: The FC Firewall offers an API to set the Firewall State to allow for dynamic firewall behaviour based on the current vehicle state (e.g. driving, parking, in a diagnostic session). More details can be found in Section 7.3.3. Furthermore, the FC Firewall supports also the intrusion detection system by raising security events. The architecture of the FC Firewall can hence be represented as

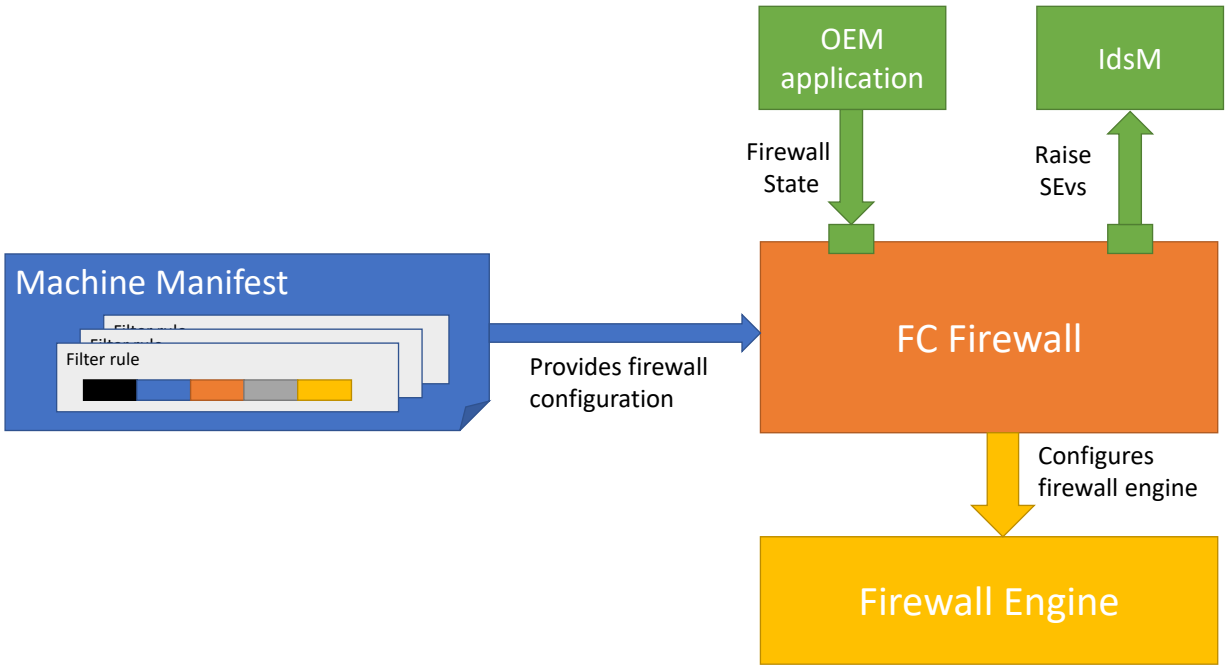


Figure 7.2: Architecture of the FC Firewall

This chapter is structured as follows:

- Sec. 7.5 describes the lifecycle of the FC Firewall
- Sec. 7.2 describes the network packet inspection, i.e. the pattern-matching part of the FirewallRules
- Sec. 7.3 describes the filtering aspect of the Firewall, i.e. which actions to carry out in case of a pattern match. This section also contains the use-cases of rate limiting and filtering based on the vehicle state
- Sec. 7.4 describes the management of Firewall rules, i.e., how to add/remove/change rules and (de-)activate rules during runtime
- Sec. 7.6.1 describes the security events raised by the Firewall

7.2 Network packet inspection

The FC Firewall manages a list of firewall rules, which consist of an expected network packet pattern and actions to be carried out in case of a pattern match. The firewall rules are modeled as FirewallRules in the AUTOSAR methodology. For every network packet that passes the network stack, the firewall compares the network packet with all configured expected patterns and carries out the action associated with the FirewallRule in case of a pattern match. The FirewallRules are ordered based on the Metamodel configuration and the firewall shall iterate through the FirewallRules in the configured order until the first pattern match.

[AP_SWS_Fw_30001] ⌈The firewall shall inspect every network packet and compare it against the ordered list of expected patterns defined in FirewallRules. In case of a pattern match, the firewall stops with the comparison against the expected patterns and carries out the action associated with the matching rule.⌋

The possible actions in case of a pattern match are described in Sec. 7.3.

The firewall supports different filtering mechanisms:

- **Stateless filtering:** Inspection of field values (e.g. header fields) and comparison against statically defined values
- **Stateful filtering:** Filtering on specific aspects of the stateful nature of the underlying protocol (e.g. allowed state transitions, number of open connections)
- **Deep packet inspection:** Inspection of application layer protocols (e.g. SOME/IP, DDS, DoIP). This can also include generic inspection of the network packet payload based on offset and expected value

The firewall performs the inspection on the complete network packet. Hence, the pattern description is comprised of expected patterns for different protocols. This is modeled by individual configuration parts for every OSI Layer (DataLinkLayerRule, NetworkLayerRule, TransportLayerRule etc.) that are aggregated by FirewallRules in the AUTOSAR Metamodel.

[AP_SWS_Fw_31001] ⌈For ingress traffic, only FirewallRules that are referenced by FirewallRuleProps.matchingIngressRule shall be considered for network traffic inspection.⌋

[AP_SWS_Fw_31002] ⌈For egress traffic, only FirewallRules that are referenced by FirewallRuleProps.matchingEgressRule shall be considered for network traffic inspection.⌋

[AP_SWS_Fw_30002] ⌈A FirewallRule is considered a match if all aggregated DataLinkLayerRules, NetworkLayerRules, TransportLayerRules, SomeipProtocolRules, SomeipSdRules, DdsRules, DoIpRules and PayloadBytePatternRules generate a match for their respective protocol.⌋

7.2.1 Stateless packet inspection

For stateless packet inspection, the firewall inspects the network protocol headers up to OSI layer 4 and compares them against expected values.

[AP_SWS_Fw_30003] ⌈The firewall shall compare the expected values defined in DataLinkLayerRule of every FirewallRule against the header fields in the network packet. If all values match, the DataLinkLayerRule is considered a match. Otherwise the DataLinkLayerRule is considered a no-match⌋

[AP_SWS_Fw_30004] ⌈The firewall shall compare the expected values defined in NetworkLayerRule of every FirewallRule against the header fields in the network packet. If all values match, the NetworkLayerRule is considered a match. Otherwise the NetworkLayerRule is considered a no-match⌋

[AP_SWS_Fw_30005] ⌈The firewall shall compare the expected values defined in TransportLayerRule of every FirewallRule against the header fields in the network packet. If all values match, the TransportLayerRule is considered a match. Otherwise the TransportLayerRule is considered a no-match⌋

The firewall shall only inspect the parameters that were configured within a FirewallRule. Parameters that are available within the Metamodel but are not configured shall be ignored.

In some cases, it is useful to not limit the expected pattern to specific values, but to also allow for values to be in a specific range. Ranges can either be defined by subnets (e.g. for MAC and IP addresses) or by defining the minimal and maximal value of the parameter (e.g. for ports).

[AP_SWS_Fw_30006] ⌈If a DataLinkLayerRule defines a subnet by means of DataLinkLayerRule.sourceMacAddressMask or DataLinkLayerRule.destinationMacAddressMask, all addresses within the network packet that fall within this subnet are considered a match for this DataLinkLayerRule⌋

[AP_SWS_Fw_30007] ⌈If an Ipv4Rule defines a subnet by means of Ipv4Rule.sourceNetworkMask or Ipv4Rule.destinationNetworkMask, all addresses within the network packet that fall within this subnet are considered a match for this Ipv4Rule⌋

[AP_SWS_Fw_30008] ⌈If an Ipv6Rule defines a subnet by means of Ipv6Rule.sourceNetworkMask or Ipv6Rule.destinationNetworkMask, all addresses within the network packet that fall within this subnet are considered a match for this Ipv6Rule⌋

[AP_SWS_Fw_30009] ⌈If an Ipv4Rule defines a range by means of Ipv4Rule.ttlMin and Ipv4Rule.ttlMax, all values within the network packet that fall within this range (including the minimal and maximal value) are considered a match for this Ipv4Rule⌋

[AP_SWS_Fw_30010] ⌈If a TransportLayerRule defines a range by means of TransportLayerRule.minSourcePortNumber and TransportLayerRule.maxSourcePortNumber or by means of TransportLayerRule.minDestinationPortNumber and TransportLayerRule.maxDestinationPortNumber, all values within the network packet that fall within this range (including the minimal and maximal value) are considered a match for this TransportLayerRule⌋

The firewall shall also be able to verify if the checksum of the respective protocol is valid.

[AP_SWS_Fw_30011] ⌈If Ipv4Rule.checksumVerification, IcmpRule.checksumVerification or TransportLayerRule.checksumVerification is set to true, the firewall shall check if the checksum field for the respective protocol is available in the network packet. If the checksum is available, the respective Ipv4Rule, IcmpRule or TransportLayerRule is considered a match.⌋

7.2.1.1 Inspection of not modeled protocols

For stateless packet inspection, the FC Firewall natively supports the modeled protocols Ethernet, IPv4, IPv6, ICMP, TCP and UDP. Additional protocols can be added by two mechanisms:

EtherType inspection: Many protocols can already be identified on data link layer by means of the EtherType (as defined in IEEE 802.3 [5]). These protocols can therefore be blocked by the FC Firewall by configuring DataLinkLayerRule.etherType within a FirewallRule. Examples for protocols that can be identified based on EtherTypes can be found in Table 7.1.

EtherType	*Protocol*
0x0806	Address Resolution protocol over IPv4 (ARP)
0x22EA	Stream Reservation Protocol (SRP)
0x22F0	Audio Video Transport Protocol (AVTP)
0x88E5	MACsec
0x88F7	Precision Time Protocol (PTP) over IEEE 802.3 Ethernet
0xF1C1	Redundancy Tag (as defined in IEEE 802.1CB Frame Replication and Elimination for Reliability)

Table 7.1: EtherType examples

Generic inspection based on byte pattern: The FC Firewall supports generic inspection of network packets based on expected byte-values at given offsets. This feature is specified in Sec. 7.2.3.4 and allows for detailed inspection of protocols that are not modeled within the FC Firewall as well as inspection of payload data.

7.2.2 Stateful packet inspection

In stateful packet inspection, the FC Firewall takes into account the stateful nature of TCP and performs additional checks to identify timeouts, limit the number of open connections and perform checks against the TCP statemachine.

[AP_SWS_Fw_30012] ⌈If the parameter TcpRule.timeoutCheck is set, the firewall shall store the time of the latest network packet for the respective communication peer. If the time between the latest and current network packet is smaller than the value of TcpRule.timeoutCheck, the TcpRule is considered a match.⌋

[AP_SWS_Fw_30013] ⌈If the parameter TcpRule.numberOfParallelTcpSessions is set, the firewall shall keep track of the number of open TCP connections. If a network packet wants to open a new TCP session and the number of open TCP sessions including the newly opened TCP session is smaller than TcpRule.numberOfParallelTcpSessions, the TcpRule is considered a match.⌋

[AP_SWS_Fw_30014] ⌈If the parameter TcpRule.stateManagementBasedOnTcpFlags is set to true, the firewall shall check whether the network packet wants to perform an allowed TCP state transition according to RFC 793. If this state transition is allowed, the TcpRule is considered a match.⌋

7.2.3 Deep packet inspection

The firewall also supports inspection of application layer protocols to perform deep packet inspection of network packets. To this end, the firewall supports deep packet inspection of the following protocols:

- SOME/IP (including SOME/IP-SD)
- DDS
- DoIP
- Generic deep packet inspection

7.2.3.1 SOME/IP

For SOME/IP [6] the inspection focuses on the SOME/IP header fields. The header fields also include service-specific information like Service ID, Method ID etc., so the deep packet inspection of SOME/IP packets can be used to perform access control to individual services.

It is possible that multiple SOME/IP messages are transported within one TCP frame. Within the FC Firewall metamodel, every FirewallRule can aggregate at most

one SOME/IP message. If a network packet contains more than one SOME/IP message, the FC Firewall has thus to check that for every SOME/IP message within the network packet a valid FirewallRule exists.

[AP_SWS_Fw_30015] ⌈If the network packet to be inspected contains one or multiple SOME/IP messages, the FC Firewall shall find the subset of FirewallRules, where the respective DataLinkLayerRule, NetworkLayerRule and TransportLayerRule have provided a match and a SomeipProtocolRule is aggregated.⌋

[AP_SWS_Fw_30016] ⌈For this subset, the FC Firewall shall compare their expected values against the SOME/IP header fields of the SOME/IP messages in the network packet. If all values match and if for all FirewallRules the FirewallRuleProps.action from the referenced FirewallRuleProps is the same, the respective FirewallRules are considered to be matches.⌋

Additionally, the FC Firewall supports length verification, i.e. to check whether the TCP payload length matches the combined length of all included SOME/IP messages

[AP_SWS_Fw_30017] ⌈If the parameter SomeipProtocolRule.lengthVerification is set to true, the firewall shall compare the TCP payload size with the cumulative length of all included SOME/IP messages. If both values match, the SomeipProtocolRule is considered a match. Otherwise the SomeipProtocolRule is considered a no-match⌋

The FC Firewall also supports inspection of the SOME/IP service discovery protocol [7]. Similar to regular SOME/IP inspection, it is also possible to group multiple SOME/IP-SD messages within one network packet. Hence, the FC Firewall implements a similar logic to inspect network packets with multiple SOME/IP-SD messages.

[AP_SWS_Fw_30018] ⌈If the network packet to be inspected contains one or multiple SOME/IP-SD messages, the FC Firewall shall find the subset of FirewallRules, where the respective DataLinkLayerRule, NetworkLayerRule and TransportLayerRule have provided a match and a SomeipSdRule is aggregated.⌋

[AP_SWS_Fw_30019] ⌈For this subset, the FC Firewall shall compare their expected values against the SOME/IP-SD header fields of the SOME/IP-SD messages in the network packet. If all values match and if for all FirewallRules the FirewallRuleProps.action from the referenced FirewallRuleProps is the same, the respective FirewallRules are considered to be matches.⌋

[AP_SWS_Fw_30020] ⌈If a SomeipSdRule is aggregated in a FirewallRule, the firewall shall compare the SOME/IP header fields of all SOME/IP-SD messages within the network packet against the default values defined in PRS_SOMEIPServiceDiscoveryProtocol [7]. If all values match, the SomeipSdRule is considered a match. Otherwise the SomeipSdRule is considered a no-match⌋

Similar to the stateless network packet inspection on lower layers, it is also possible to define ranges of allowed values by using minimal and maximal values. In case such

a range is defined, all values from the network packet that fall within this range are a match

[AP_SWS_Fw_30021] ⌈If a SomeipSdRule defines a range by means of SomeipSdRule.minMinorVersion and SomeipSdRule.maxMinorVersion or by means of SomeipSdRule.minMajorVersion and SomeipSdRule.maxMajorVersion, all values within the network packet that fall within this range (including the minimal and maximal value) are considered a match for this SomeipSdRule⌋

7.2.3.2 DDS

Deep packet inspection of DDS messages is based on the DDS Interoperability Wire Protocol (DDS-RTPS [8]), which specifies the representation of DDS messages within network packets: DDS-RTPS defines a packet format that consists of a RTPS header and multiple RTPS submessages that can be accumulated within one RTPS message. Additionally, DDS allows also for multiple RTPS messages within one TCP or UDP packet. In analogy to SOME/IP, the FC Firewall allows only the configuration of a single RTPS header and submessage within a FirewallRule and the FC Firewall has hence to compare the network packet against all configured RTPS rules.

[AP_SWS_Fw_30022] ⌈If the network packet to be inspected contains one or multiple DDSI-RTPS messages, the FC Firewall shall find the subset of FirewallRules, where the respective DataLinkLayerRule, NetworkLayerRule and TransportLayerRule have provided a match and a DdsRule is aggregated.⌋

[AP_SWS_Fw_30023] ⌈For this subset, the FC Firewall shall compare their expected values against the fields of the DDS-RTPS messages and submessages in the network packet. If all values match and if for all FirewallRules the FirewallRuleProps.action from the referenced FirewallRuleProps is the same, the respective FirewallRules are considered to be matches.⌋

7.2.3.3 DoIP

The FC Firewall supports deep packet inspection of DoIP messages [9], where the firewall inspects the DoIP header as well as parts of the payload (DoIP source/destination address, UDS services). The FC Firewall does not, however, perform deep packet inspection of the UDS protocol, i.e., inspection on the level of individual DIDs, RIDs etc. Nevertheless, these kind of checks are still possible to implement by means of the generic inspection feature described in Sec. 7.2.3.4.

[AP_SWS_Fw_30024] ⌈The firewall shall compare the expected values defined in DoIpRule of every FirewallRule against the DoIP header fields in the network packet. If all values match, the DoIpRule is considered a match. Otherwise the DoIpRule is considered a no-match⌋

Similar to the stateless network packet inspection on lower layers, it is also possible to define ranges of allowed values by using minimal and maximal values. In case such a range is defined, all values from the network packet that fall within this range are a match

[AP_SWS_Fw_30025] ⌈If a DoIpRule defines a range by means of DoIpRule.sourceMinAddress and DoIpRule.sourceMaxAddress or by means of DoIpRule.destinationMinAddress and DoIpRule.destinationMaxAddress, all values within the network packet that fall within this range (including the minimal and maximal value) are considered a match for this DoIpRule⌋

7.2.3.4 Generic inspection

The FC Firewall allows for generic inspection of the network packets (e.g. to perform payload inspection or to inspect protocols that are not natively supported by the firewall). To this end, every FirewallRule can aggregate multiple PayloadBytePatternRules, which specify the expected byte values at a specific offset within the network packet.

[AP_SWS_Fw_30026] ⌈The firewall shall compare the expected values defined in the PayloadBytePatternRules of every FirewallRule against the values at the specified offsets in the network packet. If all values match, the PayloadBytePatternRules are considered matches.⌋

7.3 Network packet filtering

After describing the rule-based network packet inspection process based on pattern-matching in chapter 7.2, this chapter specifies the associated filtering mechanisms supported by the FC Firewall. Section 7.3.1 describes the pattern-matching-based filtering approach using Allowlists and Blocklists, section 7.3.2 specifies the rate limiting feature of the FC Firewall and section 7.3.3 outlines the state-dependent filtering mechanism based on configurable Firewall States.

7.3.1 Allowlists and Blocklists

Firewalls can generally be categorized into two groups: Allowlist and Blocklist firewalls. In an Allowlist firewall, all network traffic that is allowed to pass the firewall is specified (i.e. patterns are defined), all network packets without a matching pattern are blocked. Blocklist firewalls implement the inverse approach: Only explicitly defined network packets are blocked, whereas traffic without a matching pattern is allowed to pass the firewall.

The action to be carried out in the case of a match of a FirewallRule is defined by the parameter FirewallRuleProps.action in the referenced FirewallRuleProps.

[AP_SWS_Fw_40001] ⌈If a FirewallRule is a match and FirewallRuleProps.action in the referenced FirewallRuleProps is set to allow, the firewall shall allow the network packet to continue its flow within the network stack⌋

[AP_SWS_Fw_40002] ⌈If a FirewallRule is a match and FirewallRuleProps.action in the referenced FirewallRuleProps is set to block, the firewall shall drop the network packet⌋

In addition, it has to be defined how the Firewall shall behave in the case that no FirewallRule generated a match:

[AP_SWS_Fw_40003] ⌈If no FirewallRule matches the network packet, the firewall shall drop the network packet if StateDependentFirewall.defaultAction is set to block and let it pass if it is set to allow.⌋

The FC Firewall allows also for mixed Allow-/Blocklist Firewalls: it is possible to define FirewallRules that block a network packet upon a pattern match together with FirewallRules that allow a network packet to pass upon a pattern match. This seems redundant at first, since network packets that provide no match are caught by the Firewalls default behaviour, but there is one specific reason for this design: The explicit definition of network packet patterns allows for the usage of the pattern matching algorithm, which in turn allows for a dedicated mapping of IDS security events for these network packets. See Sec. 7.6.1 for more details.

7.3.2 Rate limiting

The firewall supports rate limiting based on the pattern matching algorithm to identify off-frequency cyclic messages, that can be caused by, e.g., a man-in-the-middle attack or a faulty ECU. To realize this, the FC Firewall implements the leaky bucket algorithm, which is also supported on HW side by some products.

[AP_SWS_Fw_40004] ⌈If the parameters FirewallRule.bucketSize and FirewallRule.refillAmount are configured for a FirewallRule, the FC Firewall shall keep track of the number of pattern matches by means of a leaky bucket algorithm, where FirewallRule.refillAmount defines the decrement rate of the leaky bucket algorithm and the counter is increased by one for every pattern match⌋

[AP_SWS_Fw_40005] ⌈In the case of a pattern match and if the leaky bucket counter is bigger than FirewallRule.bucketSize, the firewall shall drop the network packet.⌋

7.3.3 State dependent filtering

The in-vehicle traffic can strongly depend on the vehicle's situation (e.g. driving, parking, in a diagnostic session etc.), which also renders the expected network packets to be different depending on the current vehicle state. The FC Firewall supports this use-case by being state-dependent: FirewallRules can be associated with specific Firewall States, that are pre-configured on a project-specific basis by the integrator and that can be managed by a user application. Within the AUTOSAR Meta Model, this feature is realized by StateDependentFirewalls that aggregate a set of FirewallRules. Only one of the StateDependentFirewalls can be active, which means that only the FirewallRules associated with that StateDependentFirewall are active

[AP_SWS_Fw_40006] ⌈The FC Firewall shall ensure that only one StateDependentFirewall is active⌋

[AP_SWS_Fw_40007] ⌈Only the FirewallRules referenced by the currently active StateDependentFirewall shall be taken into account for the network packet inspection. FirewallRules that are not referenced by the currently active StateDependentFirewall shall be ignored⌋

[AP_SWS_Fw_40008] ⌈For no-match cases, the StateDependentFirewall.defaultAction defined in the currently active StateDependentFirewall shall be used⌋

The FC Firewall provides the ara::fw::FirewallStateSwitchInterface API to switch the currently active StateDependentFirewall.

[AP_SWS_Fw_40013] FirewallStateSwitchInterface access control ⌈The FC Firewall shall allow a process to update the currently active StateDependentFirewall using ara::fw::FirewallStateSwitchInterface::SwitchFirewallState API, only if a AdaptiveFirewallToPortPrototypeMapping exists that links

- The process, that is requesting the Firewall State update.
- The RPortPrototype, typed by a FirewallMode SwitchInterface (for more details, please refer to AdaptiveFirewallToPortPrototypeMapping).

⌋

[AP_SWS_Fw_40009] ⌈If a ModeDeclaration is reported to the FC Firewall by means of ara::fw::FirewallStateSwitchInterface::SwitchFirewallState, the referenced StateDependentFirewall shall be considered as active.⌋

[AP_SWS_Fw_40010] ⌈If a ModeDeclaration is reported to the FC Firewall by means of ara::fw::FirewallStateSwitchInterface::SwitchFirewallState and the referenced StateDependentFirewall is empty

(i.e. not configured or no FirewallRuleProps aggregated), the FC Firewall shall keep the currently active StateDependentFirewall and return ara::fw::FirewallStateSwitchInterface::SwitchFirewallState.kServiceNotAvailable.⌋

[AP_SWS_Fw_40011] ⌈If no ModeDeclaration has been reported to the FC Firewall, the FC Firewall shall consider the StateDependentFirewall as active where the referenced ModeDeclaration is referenced as initialMode by the ModeDeclarationGroup.⌋

[AP_SWS_Fw_40012] ⌈If the ara::fw::FirewallStateSwitchInterface::SwitchFirewallState API is called and the FC Firewall has lost the connection to the daemon that runs the firewall engine, the FC Firewall shall return ara::fw::FirewallStateSwitchInterface::SwitchFirewallState.kServiceNotAvailable.⌋

7.4 Firewall Rule Management

After their initial deployment, the FirewallRules need to be managed to address certain changes within the lifetime of the vehicle, e.g. newly deployed applications that should be added to the Allowlist or changes in the threat landscape that would require specific network packets to be blocked. While the first example can be thoroughly planned and rolled out over a longer time, the latter one might be more pressing, e.g. if an attacker is currently attacking the vehicle, a newly added block rule could help mitigating the attack. The FC Firewall supports two ways of managing FirewallRules: By performing an OTA update or by (de-)activating FirewallRules during runtime (not supported in this release).

The FC Firewall configuration including the FirewallRules are deployed to the AUTOSAR Adaptive Platform by means of the respective manifests. Hence, in order to add new rules, change existing ones or remove them completely, the firewall configuration can be updated by means of an OTA update using UCM. This is the preferred way of adding new rules that account for newly deployed applications, for instance, that require a new allow rule. Since these applications are also installed using UCM, it is recommended to add the changed firewall configuration to the vehicle update campaign.

As an alternative way, the FC Firewall also offers an interface that allows to dynamically activate or deactivate FirewallRules during runtime. This interface can be used by an Intrusion Prevention System to manage the available FirewallRules, e.g. to block malicious communication by activating a block rule or deactivating an allow rule. The interface can only be used to manage already configured firewall rules, new rules can only be deployed using the OTA mechanism described above.

For this release, only the management mechanism via UCM is supported. The management mechanisms for (de-)activating individual rules during runtime is not supported for this release.

7.5 Functional cluster life-cycle

Using `ara::core::Initialize` and `ara::core::Deinitialize`, the application can initialize and deinitialize the `FC Firewall`.

Applications are expected not to call any API of the `FC Firewall` before `ara::core::Initialize` or after `ara::core::Deinitialize`.

7.5.1 Startup

[AP_SWS_Fw_00001] ⌈When `ara::core::Initialize` is called, the `FC Firewall` shall read in the manifest information and prepare the access structures necessary to communicate with applications.⌋

Access structures may encompass the communication channel between the application process and the stack process (if there is any) or other resource required by the firewall.

7.5.2 Shutdown

[AP_SWS_Fw_00002] ⌈When `ara::core::Deinitialize` is called, the `FC Firewall` shall close all acquired handles and free all access structures.⌋

7.5.3 Daemon crash

No content.

7.6 Reporting

7.6.1 Security Events

Firewalls are a crucial part of Intrusion Detection Systems (`IDS`), as they are monitoring the complete network traffic and are thus able to identify attacks within the in-vehicle network. AUTOSAR specifies the vehicle part of an `IDS` within the `IdsM` (IDS Manager), which aggregates and qualifies security events raised by IDS sensors and forwards them to the configured sink, either the persistent memory or the vehicle-central IDS instance (`IdsR` in the AUTOSAR IDS concept).

The `FC Firewall` supports the `IDS` by acting as an IDS sensor and raising security events (`SEv`s) to the `IdsM`. To this end, the `FC Firewall` specifies a set of `SEv`s (see Sec. 7.6.1.1) as well as conditions on when to raise them (see Sec. 7.6.1.2).

7.6.1.1 SEvs raised by the firewall

The IdsM specifies SEvs to consist of a unique SEv ID and associated context data, that provides more details about the nature of the incident. The IdsM qualifies these SEvs by running them through a filter chain. During this process, the IdsM can also aggregate multiple SEvs with the same SEv IDs, where only the context data of one SEv is kept. This behaviour can cause information loss and needs to be reflected when designing the SEvs raised by the FC Firewall - the SEvs need to be fine-grained enough to limit information loss as much as possible while still being precise and clear in their specification. To this end, the FC Firewall specifies a set of SEvs that is focusing on the individual protocols that are inspected by the firewall:

[AP_SWS_Fw_61000] Security events for firewall (AP) ⌈

Name	*Description*	*ID*
SEV_FW_PACKET_BLOCKED_IPV4_MISMATCH	A network packet was blocked due to a rule mismatch on IPv4 layer.	51
SEV_FW_PACKET_BLOCKED_IPV6_MISMATCH	A network packet was blocked due to a rule mismatch on IPv6 layer.	52
SEV_FW_PACKET_BLOCKED_ICMP_MISMATCH	A network packet was blocked due to a rule mismatch within the ICMP protocol.	53
SEV_FW_PACKET_BLOCKED_TCP_MISMATCH	A network packet was blocked due to a rule mismatch on TCP layer.	54
SEV_FW_PACKET_BLOCKED_UDP_MISMATCH	A network packet was blocked due to a rule mismatch on UDP layer.	55
SEV_FW_PACKET_BLOCKED_SOMEIP_MISMATCH	A network packet was blocked due to a rule mismatch in the SOME/IP protocol.	56
SEV_FW_PACKET_BLOCKED_SOMEIPSD_MISMATCH	A network packet was blocked due to a rule mismatch in the SOME/IP SD protocol.	57
SEV_FW_PACKET_BLOCKED_DDS_MISMATCH	A network packet was blocked due to a rule mismatch in the DDS-RTPS protocol.	58
SEV_FW_PACKET_BLOCKED_DOIP_MISMATCH	A network packet was blocked due to a rule mismatch in the DoIP protocol.	59
SEV_FW_PACKET_BLOCKED_GENERIC_MISMATCH	A network packet was blocked due to a rule mismatch on generic inspection level.	60
SEV_FW_PACKET_BLOCKED_TCP_MAXCONNECTIONS	A network packet was blocked due to the maximal number of open TCP connections was reached.	61
SEV_FW_PACKET_BLOCKED_TCP_TIMEOUT	A network packet was blocked due to TCP timeout.	62
SEV_FW_PACKET_BLOCKED_TCP_STATETRANSITION	A network packet was blocked due to an invalid TCP state transition.	63
SEV_FW_PACKET_BLOCKED_RATELIMIT	A network packet was blocked due to the rate limit was reached.	64
SEV_FW_PACKET_BLOCKED_DATALINKLAYER_MISMATCH	A network packet was blocked due to a rule mismatch on data link layer.	77
SEV_ACCESS_CONTROL_FIREWALL_IAM_ACCESS_DENIED	Access of an application to a resource provided by the firewall was denied.	131

⌋

[AP_SWS_Fw_60001] Security event context data definition: SEV_FW_PACKET_BLOCKED_DATALINKLAYER_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_DATALINKLAYER_MISMATCH*	
ID	77	
Description	A network packet was blocked due to a rule mismatch on data link layer.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to • CP: FwSEvEthernetFrameMaxLength • AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60020] Security event context data definition: SEV_FW_PACKET_BLOCKED_IPV4_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_IPV4_MISMATCH*	
ID	51	
Description	A network packet was blocked due to a rule mismatch on IPv4 layer.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60021] Security event context data definition: SEV_FW_PACKET_BLOCKED_IPV6_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_IPV6_MISMATCH*	
ID	52	
Description	A network packet was blocked due to a rule mismatch on IPv6 layer.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array

▽

△

SEV Name	*SEV_FW_PACKET_BLOCKED_IPV6_MISMATCH*	
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60022] Security event context data definition: SEV_FW_PACKET_BLOCKED_ICMP_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_ICMP_MISMATCH*	
ID	53	
Description	A network packet was blocked due to a rule mismatch within the ICMP protocol.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60023] Security event context data definition: SEV_FW_PACKET_BLOCKED_TCP_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_TCP_MISMATCH*	
ID	54	
Description	A network packet was blocked due to a rule mismatch on TCP layer.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60024] Security event context data definition: SEV_FW_PACKET_BLOCKED_UDP_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_UDP_MISMATCH*	
ID	55	
Description	A network packet was blocked due to a rule mismatch on UDP layer.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60025] Security event context data definition: SEV_FW_PACKET_BLOCKED_SOMEIP_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_SOMEIP_MISMATCH*	
ID	56	
Description	A network packet was blocked due to a rule mismatch in the SOME/IP protocol.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60026] Security event context data definition: SEV_FW_PACKET_BLOCKED_SOMEIPSD_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_SOMEIPSD_MISMATCH*	
ID	57	
Description	A network packet was blocked due to a rule mismatch in the SOME/IP SD protocol.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract

▽

△

SEV Name	*SEV_FW_PACKET_BLOCKED_SOMEIPSD_MISMATCH*	
	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.

⌋

[AP_SWS_Fw_60027] Security event context data definition: SEV_FW_PACKET_BLOCKED_DDS_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_DDS_MISMATCH*	
ID	58	
Description	A network packet was blocked due to a rule mismatch in the DDS-RTPS protocol.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60028] Security event context data definition: SEV_FW_PACKET_BLOCKED_DOIP_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_DOIP_MISMATCH*	
ID	59	
Description	A network packet was blocked due to a rule mismatch in the DoIP protocol.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60029] Security event context data definition: SEV_FW_PACKET_BLOCKED_GENERIC_MISMATCH ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_GENERIC_MISMATCH*	
ID	60	
Description	A network packet was blocked due to a rule mismatch on generic inspection level.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

Additionally, the FC Firewall also specifies a set of SEvs that are focusing on the stateful properties of TCP connections:

[AP_SWS_Fw_60002] Security event context data definition: SEV_FW_PACKET_BLOCKED_TCP_MAXCONNECTIONS ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_TCP_MAXCONNECTIONS*	
ID	61	
Description	A network packet was blocked due to the maximal number of open TCP connections was reached.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60030] Security event context data definition: SEV_FW_PACKET_BLOCKED_TCP_TIMEOUT ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_TCP_TIMEOUT*	
ID	62	
Description	A network packet was blocked due to TCP timeout.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array

▽

△

SEV Name	*SEV_FW_PACKET_BLOCKED_TCP_TIMEOUT*	
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

[AP_SWS_Fw_60031] Security event context data definition: SEV_FW_PACKET_BLOCKED_TCP_STATETRANSITION ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_TCP_STATETRANSITION*	
ID	63	
Description	A network packet was blocked due to an invalid TCP state transition.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	Received EthernetFrame, truncated to the first bytes according to - CP: FwSEvEthernetFrameMaxLength - AP: maxLength of the ContextDataElement Ethernet Frame from the SecurityExtract
	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.

⌋

Finally, a separate SEv is defined for network packets that are dropped due to the rate limiting feature:

[AP_SWS_Fw_60003] Security event context data definition: SEV_FW_PACKET_BLOCKED_RATELIMIT ⌈

SEV Name	*SEV_FW_PACKET_BLOCKED_RATELIMIT*	
ID	64	
Description	A network packet was blocked due to the rate limit was reached.	
Context Data Version	2	
Context Data	*Data Type*	*Allowed Values*
MAC_Address	uint8 [6]	

⌋

For denied access to FC Firewall resources the following SEv is defiend:

[AP_SWS_Fw_60032] Security event context data definition: SEV_ACCESS_CONTROL_FIREWALL_IAM_ACCESS_DENIED ⌈

SEV Name	*SEV_ACCESS_CONTROL_FIREWALL_IAM_ACCESS_DENIED*	
ID	131	
Description	Access of an application to a resource provided by the firewall was denied.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
UserId	uint32	

⌋

7.6.1.2 Raising SEvs

With regards to the general pattern matching process, the FC Firewall can raise SEvs in two cases: Either the network packet does not match any FirewallRule and the default action is performed or the network packet matches a defined FirewallRule and the respective action is performed. In this release, SEvs are only raised in the first case, i.e. if no FirewallRule matches. The second case will be added in a later release. In the no-match case, SEvs make only sense when the firewall is configured to block unspecified network packets as default action.

In this case, the FC Firewall has to identify on which network protocol the violation occurred to raise the corresponding SEv. To this end, the FC Firewall has to identify the rule that fits the no-matched network packet best by calculating the least distance as follows:

[AP_SWS_Fw_60004] ⌈If a network packet is blocked by the default action, the FC Firewall shall identify the network protocol that was not matching the FirewallRules. To this end, the FC Firewall shall iterate over all FirewallRules and identify the rules for which the most succeeding protocols starting from the lowest ISO OSI Layer and going the ISO OSI Layer upwards are matching the network packet. The protocol on the next OSI OSI Layer is the network protocol that is considered not to match the FirewallRules.⌋

The following example illustrates the mechanism

Protocol *Field*	**IP** *IP addr*	**TCP** *Port*	**SOME/IP** *Service ID*
Network Packet	1.2.3.4	1000	0xABCD
FW Rule #1	1.2.3.4	1000	0x1234
FW Rule #2	1.2.3.4	1000	0x3456
FW Rule #3	1.2.3.4	2000	0x5678
FW Rule #4	5.6.7.8	3000	0x5678
FW Rule #5	5.6.7.8	3000	0xABCD

Figure 7.3: SEV protocol matching process

The incoming network packet matches none of the defined rules, so the default action applies here. The network packet matches the `Ipv4Rule` and `TcpRule` for rule number 1 and 2, only `Ipv4Rule` for rule number 3 and only `SomeipProtocolRule` for rule number 5. Rule 1 and 2 have the most succeeding matching ISO `OSI Layer`s starting from the lowest network layer (in contrast to Rule 5, for example, that has a match on SOME/IP layer but no matches on lower layers.). The rule mismatch is hence occurring on the SOME/IP layer and a `SEv` shall be raised for this protocol.

[AP_SWS_Fw_60005] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the `FirewallRule`s is Ethernet, the `FC Firewall` shall raise the `SEv` SEV_FW_PACKET_BLOCKED_DATALINKLAYER_MISMATCH to the `IdsM`.⌋

[AP_SWS_Fw_60006] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the `FirewallRule`s is IPv4, the `FC Firewall` shall raise the `SEv` SEV_FW_PACKET_BLOCKED_IPV4_MISMATCH to the `IdsM`.⌋

[AP_SWS_Fw_60007] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the `FirewallRule`s is IPv6, the `FC Firewall` shall raise the `SEv` SEV_FW_PACKET_BLOCKED_IPV6_MISMATCH to the `IdsM`.⌋

[AP_SWS_Fw_60008] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the `FirewallRule`s is ICMP, the `FC Firewall` shall raise the `SEv` SEV_FW_PACKET_BLOCKED_ICMP_MISMATCH to the `IdsM`.⌋

[AP_SWS_Fw_60009] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the `FirewallRule`s is TCP, the `FC Firewall` shall raise the `SEv` SEV_FW_PACKET_BLOCKED_TCP_MISMATCH to the `IdsM`.⌋

[AP_SWS_Fw_60010] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the FirewallRules is UDP, the FC Firewall shall raise the SEv SEV_FW_PACKET_BLOCKED_UDP_MISMATCH to the IdsM.⌋

[AP_SWS_Fw_60011] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the FirewallRules is SOME/IP, the FC Firewall shall raise the SEv SEV_FW_PACKET_BLOCKED_SOMEIP_MISMATCH to the IdsM.⌋

[AP_SWS_Fw_60012] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the FirewallRules is SOME/IP-SD, the FC Firewall shall raise the SEv SEV_FW_PACKET_BLOCKED_SOMEIPSD_MISMATCH to the IdsM.⌋

[AP_SWS_Fw_60013] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the FirewallRules is DDS, the FC Firewall shall raise the SEv SEV_FW_PACKET_BLOCKED_DDS_MISMATCH to the IdsM.⌋

[AP_SWS_Fw_60014] ⌈If a network packet is blocked by the default action and the network protocol that was not matching the FirewallRules is DoIP, the FC Firewall shall raise the SEv SEV_FW_PACKET_BLOCKED_DOIP_MISMATCH to the IdsM.⌋

[AP_SWS_Fw_60015] ⌈If a network packet is blocked by the default action and no network protocol that was not matching the FirewallRules could be identified (e.g. because there was a mismatch in the payload using a PayloadBytePatternRule), the FC Firewall shall raise the SEv SEV_FW_PACKET_BLOCKED_GENERIC_MISMATCH to the IdsM.⌋

In addition to pattern mismatches, the FC Firewall shall also raise SEvs for network packets that have been blocked due to the stateful nature of TCP

[AP_SWS_Fw_60016] ⌈If a network packet is blocked due to the maximum number of connections reached (described in [AP_SWS_Fw_30013]), the FC Firewall shall raise the SEv SEV_FW_PACKET_BLOCKED_TCP_MAXCONNECTIONS to the IdsM.⌋

[AP_SWS_Fw_60017] ⌈If a network packet is blocked due to the TCP timeout filter described in [AP_SWS_Fw_30011], the FC Firewall shall raise the SEv SEV_FW_PACKET_BLOCKED_TCP_TIMEOUT to the IdsM.⌋

[AP_SWS_Fw_60018] ⌈If a network packet is blocked due to the TCP state transition filter described in [AP_SWS_Fw_30014], the FC Firewall shall raise the SEv SEV_FW_PACKET_BLOCKED_TCP_STATETRANSITION to the IdsM.⌋

Finally, network packets can also be dropped due to the rate limiting feature described in Sec. 7.3.2

[AP_SWS_Fw_60019] ⌈If a network packet is blocked due to the rate limiting feature described in [AP_SWS_Fw_40005], the `FC Firewall` shall raise the `SEv` SEV_FW_PACKET_BLOCKED_RATELIMIT to the `IdsM`.⌋

[AP_SWS_Fw_60035] Security event for access control

Status: DRAFT

⌈If access to resources was not granted by the `FC Firewall` due to [AP_SWS_Fw_40013], the `FC Firewall` shall raise the `SEv` SEV_ACCESS_CONTROL_FIREWALL_IAM_ACCESS_DENIED to the `IdsM`.⌋

[AP_SWS_Fw_60034] Firewall security event context data

Status: DRAFT

⌈When reporting a security event to the `IdsM`, the `FC Firewall` shall provide the Ethernet frame of the corresponding network packet truncated to the maxLength of the ContextDataElement EthernetFrame from the SecurityExtract as context data.⌋

7.6.2 Log Messages

Currently the `FC Firewall` does not produce Log Messages.

7.6.3 Violation Messages

Currently the `FC Firewall` has not defined any Violation Messages.

7.6.4 Production Errors

Currently the `FC Firewall` does not have defined Production Errors.

8 API specification

This chapter provides a reference of the APIs defined by this functional cluster. The API is described in the following chapters in tables. Table 8.1 explains the content that is described in such an API table.

Kind:	Defines the kind of the declaration that this API table describes. The following values are supported: • class (Declaration of a class) • function (Declaration of a member or non-member function) • struct (Declaration of a structure) • type alias (Declaration of a type alias) • enumeration (Declaration of an enumeration) • variable (Declaration of a variable)	
Port Interfaces:	States that the C++ API `class` is the related C++ API binding for the given modeled sub-class of `PortInterface`	
Header File:	Defines the header file to be included according to [AP_SWS_GENERAL_90001]	
Forwarding Header File:	Defines the forwarding header file to be included according to [AP_SWS_GENERAL_90001]	
Scope:	Defines the scope that may be a C++ `namespace` (in case of a `class` or non-member function) or a `class` declaration (in case of a member)	
Symbol:	C++ symbol name	
Thread Safety:	Defines whether a function is thread-safe, not thread-safe, or conditional according to [AP_SWS_GENERAL_13200] and [AP_SWS_GENERAL_13202]	
Syntax:	Description of C++ syntax	
Template Param:	Template parameter (0..*)	Template parameter(s) used to parameterize the template
Parameters (in):	Parameter declaration (0..*)	Parameter(s) that are passed to the function
Parameters (out):	Parameter declaration (0..*)	Parameter(s) that are returned to the caller
Return Value:	Return type	Type of the value that the function returns
Exception Safety:	Defines whether a function is exception-safe, not exception safe or conditionally exception safe	
Exceptions:	List of C++ `Exception`s that may be thrown by the function	
Violations:	List of violations that may raised by the function	
Errors:	Error type (0..*)	List of defined `ara::core::ErrorCode`s that may be returned by the function with their recoverability class defined in [RS_AP_00160]. APIs can be extended with vendor-specific error codes. These are not standardized by AUTOSAR
Description:	Brief description of the function	

Table 8.1: Explanation of an API table

8.1 PortInterface to API class binding

This table shows the `API class` binding for each `PortInterface` owned by this functional cluster and those functions taking an `ara::core::InstanceSpecifier` argument, designated to “construct” that class. These constructing functions may be any combination of special-member constructors, named constructor members or non-member factory constructors.

Port Interface	API Class / Function
FirewallStateSwitchInterface	[AP_SWS_Fw_82001] Definition of API class ara::fw::FirewallStateSwitchInterface
	[AP_SWS_Fw_82002] Definition of API function ara::fw::FirewallStateSwitchInterface::FirewallStateSwitchInterface

Table 8.2: PortInterface (sub-class) to API class / function binding

8.2 API Header Files

[AP_SWS_Fw_80001] File name, includes and multiple inclusion guard ⌈

Kind:	Header File	
Syntax:	ara/fw/states/{<fwssi-sn>}.h	
Description:	For each modeled FirewallStateSwitchInterface a header file shall be generated according to this directory and path/file name convention - a multiple inclusion guard shall be placed around the whole header file as per [AP_SWS_GENERAL_90002].	
Descriptors:	{<fwssi-sn>}	The file name as given by FirewallStateSwitchInterface.shortName

Example:	`// File=ara/fw/states/{<fwssi-sn>}.h`
	`#ifndef ARA_FW_STATES_STATE_H_`
	`#define ARA_FW_STATES_STATE_H_`
	`...`
	`#endif // ARA_FW_STATES_STATE_H_`	

⌋

8.3 API Common Data Types

[AP_SWS_Fw_81001] Definition of API enum ara::fw::states::{<fwssi-sn>}

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127

⌈

Kind:	enumeration	
Header file:	#include "ara/fw/states/{<fwssi-sn>}.h"	
Forwarding header file:	#include "ara/fw/fw_fwd.h"	
Scope:	namespace ara::fw::states	
Symbol:	{<fwssi-sn>}	
Underlying type:	std::uint32_t	
Syntax:	enum class {<fwssi-sn>} : std::uint32_t {...};	
Values:	{<fw-state-list>}	--
Description:	Defines the firewall states for the ara::fw::FirewallStateSwitchInterface	
Descriptors:	{<fw-state-list>}	*Shown as "..." in Syntax.* The list of enumerations (firewall states) for the FirewallStateSwitchInterface. For each firewall state in {<fw-state-list>}, [AP_SWS_Fw_81002] shall be applied.

⌋

[AP_SWS_Fw_81002] Definition of API variable ara::fw::states::{<symbol-fw-state>} ⌈

Kind:	variable	
Header file:	#include "ara/fw/states/{<fwssi-sn>}.h"	
Scope:	namespace ara::fw::states	
Symbol:	{<symbol-fw-state>}	
Type:	--	
Syntax:	{<symbol-fw-state>} = {<fw-state-value>};	
Description:	For each enumeration in {<fw-state-list>} in [AP_SWS_Fw_81001] there shall exist a C++ enumerator declaration.	
Descriptors:	{<symbol-fw-state>}	The firewall state enumerator symbol name as given by ModeDeclaration. shortName.
	{<fw-state-value>}	The firewall state enumerator value as given by ModeDeclaration. value. If omitted, there shall be no {<fw-state-value>} value for the enumerator.

Example:	`enum class MyFwIfStates : std::uint32_t {`
	`  kDefaultState  = 0,`
	`  kDrivingState  = 1,`
	`  kDiagnosticState = 2,`
	`};`	

⌋

8.4 API Reference

8.4.1 FirewallStateSwitchInterface

[AP_SWS_Fw_82001] Definition of API class ara::fw::FirewallStateSwitchInterface ⌈

Kind:	class	
Port Interfaces:	FirewallStateSwitchInterface	
Header file:	#include "ara/fw/firewall_state.h"	
Forwarding header file:	#include "ara/fw/fw_fwd.h"	
Scope:	namespace ara::fw	
Symbol:	FirewallStateSwitchInterface	
Syntax:	`template <typename EnumT>` `class FirewallStateSwitchInterface final {...};`	
Template param:	typename EnumT	An enum type that contains a list of firewall states
Description:	Interface to switch between firewall states, i.e., between different sets of firewall filter rules.	

⌋

[AP_SWS_Fw_82002] Definition of API function ara::fw::FirewallStateSwitchInterface::FirewallStateSwitchInterface ⌈

Kind:	function	
Header file:	#include "ara/fw/firewall_state.h"	
Scope:	`class ara::fw::FirewallStateSwitchInterface`	
Syntax:	`explicit FirewallStateSwitchInterface (const ara::core::InstanceSpecifier &instance) noexcept;`	
Parameters (in):	instance	Instance specifier of the Port typed with `FirewallStateSwitchInterface`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct from an `ara::core::InstanceSpecifier`	

⌋

[AP_SWS_Fw_82003] Definition of API function ara::fw::FirewallStateSwitchInterface::~FirewallStateSwitchInterface ⌈

Kind:	function
Header file:	#include "ara/fw/firewall_state.h"
Scope:	`class ara::fw::FirewallStateSwitchInterface`
Syntax:	`~FirewallStateSwitchInterface () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor

⌋

[AP_SWS_Fw_82004] Definition of API function ara::fw::FirewallStateSwitchInterface::FirewallStateSwitchInterface ⌈

Kind:	function
Header file:	#include "ara/fw/firewall_state.h"
Scope:	`class ara::fw::FirewallStateSwitchInterface`
Syntax:	`FirewallStateSwitchInterface (const FirewallStateSwitchInterface &se)=delete;`
Description:	Copy constructor

⌋

[AP_SWS_Fw_82005] Definition of API function ara::fw::FirewallStateSwitchInterface::operator= ⌈

Kind:	function
Header file:	#include "ara/fw/firewall_state.h"
Scope:	`class ara::fw::FirewallStateSwitchInterface`
Syntax:	`FirewallStateSwitchInterface & operator= (const FirewallStateSwitchInterface &se)=delete;`
Description:	Copy assignment constructor

⌋

[AP_SWS_Fw_82006] Definition of API function ara::fw::FirewallStateSwitchInterface::FirewallStateSwitchInterface ⌈

Kind:	function	
Header file:	#include "ara/fw/firewall_state.h"	
Scope:	`class ara::fw::FirewallStateSwitchInterface`	
Syntax:	`FirewallStateSwitchInterface (FirewallStateSwitchInterface &&se) noexcept;`	
Parameters (in):	se	The `ara::fw::FirewallStateSwitchInterface` object to be moved.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move constructor	

⌋

[AP_SWS_Fw_82007] Definition of API function ara::fw::FirewallStateSwitchInterface::operator= ⌈

Kind:	function	
Header file:	#include "ara/fw/firewall_state.h"	
Scope:	`class ara::fw::FirewallStateSwitchInterface`	
Syntax:	`FirewallStateSwitchInterface & operator= (FirewallStateSwitchInterface &&se) noexcept;`	
Parameters (in):	se	The `ara::fw::FirewallStateSwitchInterface` object to be moved.
Return value:	FirewallStateSwitch Interface &	The moved `ara::fw::FirewallStateSwitchInterface` object.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment constructor	

⌋

[AP_SWS_Fw_82008] Definition of API function ara::fw::FirewallStateSwitchInterface::SwitchFirewallState ⌈

Kind:	function	
Header file:	#include "ara/fw/firewall_state.h"	
Scope:	`class ara::fw::FirewallStateSwitchInterface`	
Syntax:	`ara::core::Future< void > SwitchFirewallState (EnumT firewallState) noexcept;`	
Parameters (in):	firewallState	The FirewallState to be set.
Return value:	ara::core::Future< void >	Void if state switch was successful, otherwise it returns one of the errors specified below
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::fw::FwErrc::kService NotAvailable	no_rollback_semantics
		Communication to Firewall daemon is broken, i.e. state is not switched
	ara::fw::FwErrc::kInvalid StateDependentFirewall	rollback_semantics

▽

△

		This firewallState is not used by any StateDependentFirewall rule-sets
Description:	Switch between firewall states	

⌋

8.4.2 FirewallErrorDomain

8.4.2.1 ara::fw::FwErrc

[AP_SWS_Fw_83001] Definition of API enum ara::fw::FwErrc

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127

⌈

Kind:	enumeration	
Header file:	#include "ara/fw/fw_error_domain.h"	
Forwarding header file:	#include "ara/fw/fw_fwd.h"	
Scope:	namespace ara::fw	
Symbol:	FwErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class FwErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kServiceNotAvailable	= 1
		Communication to Firewall daemon is broken, i.e. state is not switched
	kInvalidStateDependent Firewall	= 2
		This firewallState is not used by any StateDependentFirewall rule-sets
Description:	Defines the error codes for the `ara::fw::FwErrorDomain`	

⌋

8.4.2.2 ara::fw::GetFwErrorDomain

[AP_SWS_Fw_83002] Definition of API function ara::fw::GetFwErrorDomain

Upstream requirements: RS_AP_00120, RS_AP_00130, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/fw/fw_error_domain.h"	
Scope:	namespace ara::fw	
Syntax:	`constexpr const ara::core::ErrorDomain & GetFwErrorDomain () noexcept;`	
Return value:	const ara::core::Error Domain &	Reference to the `ara::fw::FwErrorDomain` object
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe
Description:	Returns a reference to the `ara::fw::FwErrorDomain` object

⌋

8.4.2.3 ara::fw::MakeErrorCode overload for ara::fw::GetFwErrorDomain

[AP_SWS_Fw_83003] Definition of API function ara::fw::MakeErrorCode

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/fw/fw_error_domain.h"	
Scope:	namespace ara::fw	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (ara::fw::FwErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;`	
Parameters (in):	code	Error code number.
	data	Vendor defined data associated with the error
Return value:	ara::core::ErrorCode	An `ara::core::ErrorCode` object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Creates an instance of `ara::core::ErrorCode`	

⌋

8.4.2.4 ara::fw::FwException

[AP_SWS_Fw_83004] Definition of API class ara::fw::FwException

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127

⌈

Kind:	class
Header file:	#include "ara/fw/fw_error_domain.h"
Forwarding header file:	#include "ara/fw/fw_fwd.h"
Scope:	namespace ara::fw
Symbol:	FwException
Base class:	ara::core::Exception
Syntax:	`class FwException : public ara::core::Exception {...};`
Description:	Defines a class for exceptions to be thrown by the API.

⌋

[AP_SWS_Fw_83005] Definition of API function ara::fw::FwException::FwException

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/fw/fw_error_domain.h"	
Scope:	`class ara::fw::FwException`	
Syntax:	`explicit FwException (ara::core::ErrorCode errorCode) noexcept;`	
Parameters (in):	errorCode	The error code.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constructs a new `ara::fw::FwException` containing an `ara::core::ErrorCode`	

⌋

8.4.2.5 ara::fw::FwErrorDomain

[AP_SWS_Fw_83006] Definition of API class ara::fw::FwErrorDomain

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127

⌈

Kind:	class
Header file:	#include "ara/fw/fw_error_domain.h"
Forwarding header file:	#include "ara/fw/fw_fwd.h"
Scope:	namespace ara::fw
Symbol:	FwErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class FwErrorDomain final :  public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::fw::FwErrorDomain` [AP_SWS_GENERAL_90023]
Description:	Defines a class representing the firewall error domain.

⌋

[AP_SWS_Fw_83007] Definition of API type ara::fw::FwErrorDomain::Errc

Upstream requirements: RS_AP_00120, RS_AP_00130, RS_AP_00159

⌈

Kind:	type alias
Header file:	#include "ara/fw/fw_error_domain.h"
Scope:	`class ara::fw::FwErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = FwErrc;`
Description:	Alias for the error code value enumeration

⌋

[AP_SWS_Fw_83008] Definition of API type ara::fw::FwErrorDomain::Exception

Upstream requirements: RS_AP_00120, RS_AP_00130, RS_AP_00159

⌈

Kind:	type alias
Header file:	#include "ara/fw/fw_error_domain.h"
Scope:	`class ara::fw::FwErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = FwException;`
Description:	Alias for the exception base class

⌋

[AP_SWS_Fw_83009] Definition of API function ara::fw::FwErrorDomain::FwErrorDomain

Upstream requirements: RS_AP_00120, RS_AP_00130, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/fw/fw_error_domain.h"
Scope:	`class ara::fw::FwErrorDomain`
Syntax:	`FwErrorDomain ()=delete;`
Description:	Constructs a new `ara::fw::FwErrorDomain` object

⌋

[AP_SWS_Fw_83010] Definition of API function ara::fw::FwErrorDomain::Name

Upstream requirements: RS_AP_00120, RS_AP_00130, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/fw/fw_error_domain.h"	
Scope:	`class ara::fw::FwErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	As per `ara::fw::FwErrorDomain` in [AP_SWS_GENERAL_90023]
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Retrieve the name of the error domain	

⌋

[AP_SWS_Fw_83011] Definition of API function ara::fw::FwErrorDomain::Message

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/fw/fw_error_domain.h"	
Scope:	`class ara::fw::FwErrorDomain`	
Syntax:	`const char * Message (CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	The error code number.
Return value:	const char *	The message associated with the `errorCode`
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the message associated with `errorCode`	

⌋

[AP_SWS_Fw_83012] Definition of API function ara::fw::FwErrorDomain::ThrowAsException

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/fw/fw_error_domain.h"	
Scope:	`class ara::fw::FwErrorDomain`	
Syntax:	`void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;`	
Parameters (in):	errorCode	The error to throw.
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Creates a new instance of `ara::fw::FwException` from `errorCode` and throws it. As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

9 Service Interfaces

There are no provided or required service interfaces of the `FC Firewall`.

10 Configuration

The configuration structure of `FC Firewall` is described in TPS_Manifest by `AdaptiveFirewallModuleInstantiation`. This chapter defines default values and semantic constraints for this configuration model.

10.1 Default Values

This section defines the default values for attributes defined in TPS_Manifest.

No default values defined for the `FC Firewall`.

10.2 Semantic Constraints

This section defines semantic constraints for the configuration elements of the `FC Firewall` defined in TPS_Manifest.

[AP_SWS_Fw_CONSTR_00001] Configurable Namespace for Firewall ⌈`FirewallStateSwitchInterface.namespace` shall never exist.⌋

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	**AdaptiveFirewallModuleInstantiation**			
Note	This meta-class defines the attributes for the Firewall configuration on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
stateDep Firewall	StateDependentFirewall	*	ref	Firewall rules that are defined in a firewall state. **Tags:** atp.Splitkey=stateDepFirewall

Table A.1: AdaptiveFirewallModuleInstantiation

Class	**AdaptiveFirewallToPortPrototypeMapping**			
Note	This meta-class maps the AdaptiveFirewall moduleInstantiation to the RPortPrototype that is typed by a FirewallModeSwitchInterface. **Tags:** atp.recommendedPackage=AdaptiveFirewallToPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
firewall	AdaptiveFirewallModule Instantiation	0..1	ref	Reference to the Firewall module **Tags:** atp.Splitkey=firewall
process	Process	0..1	ref	The referenced Process is supposed to define the Port Prototype typed by a FirewallStateSwitchInterface, referenced by the enclosing mapping class. If this reference does not exist, any Process using the respective API can switch the firewall state. **Tags:** atp.Splitkey=process
rPort	RPortPrototype	0..1	iref	Reference to RPortPrototype typed by a FirewallMode SwitchInterface **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPort, rPort.contextRootSw ComponentPrototype, rPort.contextComponentPrototype, rPort.targetRPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.2: AdaptiveFirewallToPortPrototypeMapping

Class	**DataLinkLayerRule**			
Note	Configuration of filter rules on the DataLink layer			
Base	*ARObject*			
Aggregated by	FirewallRule.dataLinkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationMac Address	MacAddressString	0..1	attr	Filter to match packets with the destination MAC address.
destinationMac AddressMask	MacAddressString	0..1	attr	Filter to match packets with the destination MAC address range. The destinationMacAddress with the destination MacAddressMask defines the MAC address range.
etherType	PositiveInteger	0..1	attr	Filter to match packets based on the EtherType field in the Ethernet frame. The EtherType is used to indicate which protocol is encapsulated in the payload of the frame.
sourceMac Address	MacAddressString	0..1	attr	Filter to match packets with the source MAC address.
sourceMac AddressMask	MacAddressString	0..1	attr	Filter to match packets with the source MAC address range. The sourceMacAddress with the sourceMac AddressMask defines the MAC address range.
vlanId	PositiveInteger	0..1	attr	Filter of packets with a specific VlanId.
vlanPriority	PositiveInteger	0..1	attr	Filter of packets with a specific Vlan priority.

Table A.3: DataLinkLayerRule

Class	**DdsRule**			
Note	Configuration of a DDS firewall rule			
Base	*ARObject*			
Aggregated by	FirewallRule.ddsRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
appId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the appId in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches.
hostId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the hostId in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches.
instanceId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the instanceId in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches.
majorProtocol Version	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the major ProtocolVersion in the DDSI-RTPS header matches.
minorProtocol Version	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the minor ProtocolVersion in the DDSI-RTPS header matches.
productId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the productId in the DDSI-RTPS header matches.
readerEntityId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the readerEntity ID in a DDSI-RTPS submessage matches
submessage Type	PositiveInteger	0..1	attr	Defines the allowed submessage type in the DDSI-RTPS message
vendorId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the vendorId in the DDSI-RTPS header matches.
writerEntityId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the writerEntity ID in a DDSI-RTPS submessage matches

Table A.4: DdsRule

Class	DoIpRule			
Note	Configuration of a generic firewall rule			
Base	*ARObject*			
Aggregated by	FirewallRule.doIpRule			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
destinationMax Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the destination Address is smaller or equal than destinationMaxAddress.
destinationMin Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the destination Address is greater or equal than destinationMinAddress.
inverseProtocol Version	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the inverseprotocolVersion in the DoIP header matches.
payloadLength	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the payload Length in the DoIP header matches.
payloadType	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the payloadType in the DoIP header matches.
protocolVersion	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the protocol Version in the DoIP header matches.
sourceMax Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the source Address is smaller or equal than sourceMaxAddress.
sourceMin Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the source Address is greater or equal than sourceMinAddress..
udsService	PositiveInteger	0..1	attr	Filter to match DoIP messages that contain the uds Service.

Table A.5: DoIpRule

Class	FirewallRule			
Note	Firewall Rule that defines the control information in individual packets. **Tags:** atp.recommendedPackage=FirewallRules			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bucketSize	PositiveInteger	0..1	attr	This attribute defines the capacity of the queue for rate limitation (leaky-bucket Algorithm).
dataLinkLayer Rule	DataLinkLayerRule	0..1	aggr	Configuration of rules on the Data Link Layer **Tags:** atp.Splitkey=dataLinkLayerRule
ddsRule	DdsRule	0..1	aggr	Configuration of firewall rules for DDS. **Tags:** atp.Splitkey=ddsRule
doIpRule	DoIpRule	0..1	aggr	Configuration of firewall rules for DoIP messages **Tags:** atp.Splitkey=doIpRule
networkLayer Rule	NetworkLayerRule	0..1	aggr	Configuration of rules on the Network Layer **Tags:** atp.Splitkey=networkLayerRule
payloadByte PatternRule	PayloadBytePattern Rule	*	aggr	Configuration of generic firewall rules **Tags:** atp.Splitkey=payloadBytePatternRule
refillAmount	PositiveInteger	0..1	attr	This attribute defines the output rate that describes how many packets leave the queue per second (leaky-bucket Algorithm).
someipRule	SomeipProtocolRule	0..1	aggr	Configuration of firewall rules for SOME/IP messages **Tags:** atp.Splitkey=someipRule

▽

△

Class	FirewallRule			
someipSdRule	SomeipSdRule	0..1	aggr	Configuration of firewall rules for SOME/IP Service Discovery messages **Tags:** atp.Splitkey=someipSdRule
transportLayer Rule	TransportLayerRule	0..1	aggr	Configuration of rules on the Transport Layer **Tags:** atp.Splitkey=transportLayerRule

Table A.6: FirewallRule

Class	FirewallRuleProps			
Note	Firewall rule that is defined by an action that is performed if the referenced pattern matches.			
Base	*ARObject*			
Aggregated by	StateDependentFirewall.firewallRuleProps			
Attribute	Type	Mult.	Kind	Note
action	FirewallActionEnum	0..1	attr	Action that is performed by the firewall if the matching Rule is fulfilled.
matchingEgress Rule (ordered)	FirewallRule	*	ref	This element defines an egress rule expression against which the network traffic is matched.
matching IngressRule (ordered)	FirewallRule	*	ref	This element defines an ingress rule expression against which the network traffic is matched.

Table A.7: FirewallRuleProps

Class	FirewallStateSwitchInterface			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Firewall mode. **Tags:** atp.recommendedPackage=FirewallStateSwitchPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
firewallState Machine	ModeDeclarationGroup Prototype	*	aggr	The state machine of this firewall interface. **Tags:** atp.Splitkey=firewallStateMachine, firewallState Machine.shortName

Table A.8: FirewallStateSwitchInterface

Class	IcmpRule			
Note	Configuration of filter rules for ICMP (Internet Control Message Protocol).			
Base	*ARObject*			
Aggregated by	Ipv4Rule.icmpRule, Ipv6Rule.icmpRule			
Attribute	Type	Mult.	Kind	Note
checksum Verification	Boolean	0..1	attr	Defines whether a Icmp header checksum verification is performed or not.
code	PositiveInteger	0..1	attr	Filter to match packets with the Icmp code.
type	PositiveInteger	0..1	attr	Filter to match packets with the Icmp type.

Table A.9: IcmpRule

Class	**Ipv4Rule**			
Note	Configuration of filter rules on IPv4 level.			
Base	*ARObject*, *NetworkLayerRule*			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksum Verification	Boolean	0..1	attr	Defines whether a Ipv4 header checksum verification is performed or not.
destinationIp Address	Ip4AddressString	0..1	attr	Filter to match packets with the destination IPv4 address.
destination NetworkMask	Ip4AddressString	0..1	attr	Filter to match packets with the destination IPv4 address range. The destinationIpAddress with the destination NetworkMask defines the IP address range.
differentiated ServiceCode Point	PositiveInteger	0..1	attr	Filter to match packets with a DSCP value.
doNotFragment	Boolean	0..1	attr	Filter to match packets that have the doNotFragment bit in the Header set.
explicit Congestion Notification	PositiveInteger	0..1	attr	Filter to match packets with a ECN code point.
icmpRule	IcmpRule	0..1	aggr	Configuration of filter rules for ICMP (Internet Control Message Protocol). **Tags:** atp.Splitkey=icmpRule
internetHeader Length	PositiveInteger	0..1	attr	Filter to match packets with a minimum ipv4 header length.
moreFragments	Boolean	0..1	attr	Filter to match packets that have the moreFragments flag in the Header set.
protocol	PositiveInteger	0..1	attr	Filter to match packets with a IP protocol number .
sourceIp Address	Ip4AddressString	0..1	attr	Filter to match packets with the source IPv4 address.
sourceNetwork Mask	Ip4AddressString	0..1	attr	Filter to match packets with the source IPv4 address range. The sourceIpAddress with the sourceNetwork Mask defines the IP address range.
ttlMax	PositiveInteger	0..1	attr	Filter to match packets with a maximum ttl value (TimeTo Live defines the lifetime of data on the network).
ttlMin	PositiveInteger	0..1	attr	Filter to match packets with a minimum ttl value (TimeTo Live defines the lifetime of data on the network).

Table A.10: Ipv4Rule

Class	**Ipv6Rule**			
Note	Configuration of filter rules on IPv6 level.			
Base	*ARObject*, *NetworkLayerRule*			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationIp Address	Ip6AddressString	0..1	attr	Filter to match packets with the destination IPv6 address.
destination NetworkMask	Ip6AddressString	0..1	attr	Filter to match packets with the destination IPv6 address range. The destinationIpAddress with the destination NetworkMask defines the MAC address range.
flowLabel	PositiveInteger	0..1	attr	Filter to match packets with a defined flow label.
hopLimit	PositiveInteger	0..1	attr	Filter to match packets with a minimum hop limit.

▽

△

Class	**Ipv6Rule**			
icmpRule	IcmpRule	0..1	aggr	Configuration of filter rules for ICMP (Internet Control Message Protocol). **Tags:** atp.Splitkey=icmpRule
nextHeader	PositiveInteger	0..1	attr	Filter to match packets with a defined type of an extension header.
sourceIp Address	Ip6AddressString	0..1	attr	Filter to match packets with the source IPv6 address.
sourceNetwork Mask	Ip6AddressString	0..1	attr	Filter to match packets with the source IPv6 address range. The sourceIpAddress with the sourceNetwork Mask defines the IP address range.
trafficClass	PositiveInteger	0..1	attr	Filter to match packets with a defined traffic class or priority.

Table A.11: Ipv6Rule

Class	**ModeDeclaration**			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ModeDeclarationGroup.modeDeclaration			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this Mode Declaration.

Table A.12: ModeDeclaration

Class	**ModeDeclarationGroup**			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:** atp.recommendedPackage=ModeDeclarationGroups			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
initialMode	ModeDeclaration	0..1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred. **Tags:** atp.Splitkey=initialMode
mode Declaration	ModeDeclaration	*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=modeDeclaration, modeDeclaration.short Name, modeDeclaration.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime

Table A.13: ModeDeclarationGroup

Class	***NetworkLayerRule*** (abstract)			
Note	Configuration of filter rules on the Network layer			
Base	*ARObject*			
Subclasses	Ipv4Rule, Ipv6Rule			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.14: NetworkLayerRule

Class	**PayloadBytePatternRule**			
Note	Configuration of a generic firewall rule that defines the individual bytes of a message that shall match.			
Base	*ARObject*			
Aggregated by	FirewallRule.payloadBytePatternRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
payloadByte PatternRulePart	PayloadBytePattern RulePart	*	aggr	Configuration of bytes in the message, **Tags:** atp.Splitkey=payloadBytePatternRulePart

Table A.15: PayloadBytePatternRule

Class	***PortInterface*** (abstract)			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractRawDataStreamInterface*, *AbstractSuspendToRamInterface*, *AbstractSynchronizedTimeBase Interface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, FirewallState SwitchInterface, *IdsmAbstractPortInterface*, LogAndTraceInterface, ModeSwitchInterface, Network ManagementPortInterface, *PersistencyInterface*, *PlatformHealthManagementInterface*, ServiceInterface, *StateManagementPortInterface*, TriggerInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
namespace (ordered)	SymbolProps	*	aggr	This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface. This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.16: PortInterface

Class	***Referrable*** (abstract)			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, Bsw VariableAccess, CouplingPortTrafficClassAssignment, *CppImplementationDataTypeContextTarget*, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, *HwDescription Entity*, *ImplementationProps*, ModeTransition, *MultilanguageReferrable*, NmNetworkHandle, Pnc MappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, SomeipRequiredEventGroup, Tp ConnectionIdent			
Attribute	***Type***	***Mult.***	***Kind***	***Note***



△

Class	*Referrable* (abstract)			
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** atp.Splitkey=shortNameFragment xml.sequenceOffset=-90

Table A.17: Referrable

Class	**SomeipProtocolRule**			
Note	Configuration of SOME/IP firewall rules			
Base	*ARObject*			
Aggregated by	FirewallRule.someipRule			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
clientId	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the clientId in the SOME/IP header matches.
length Verification	Boolean	0..1	attr	Defines whether length verification is performed or not.
majorVersion	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the majorVersion in the SOME/IP header matches.
messageType	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the messageType in the SOME/IP header matches.
methodId	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the methodId in the SOME/IP header matches.
protocolVersion	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the protocol Version in the SOME/IP header matches.
returnCode	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the returnCode in the SOME/IP header matches.
serviceInterface Id	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the service InterfaceId in the SOME/IP header matches.

Table A.18: SomeipProtocolRule

Class	**SomeipSdRule**			
Note	Configuration of SOME/IP Service Discovery firewall rules			
Base	*ARObject*			
Aggregated by	FirewallRule.someipSdRule			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
entryType	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the entryType in the SOME/IP header matches.
eventGroupId	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the eventGroup Id in the SOME/IP header matches.
maxMajor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Major Version in the SOME/IP header is smaller or equal than maxMajorVersion.

▽

△

Class	**SomeipSdRule**			
maxMinor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Minor Version in the SOME/IP header is smaller or equal than maxMinorVersion.
minMajor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Major Version in the SOME/IP header is greater or equal than minMajorVersion.
minMinor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Minor Version in the SOME/IP header is greater or equal than minMinorVersion.
serviceInstance Id	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the service InstanceId in the SOME/IP header matches.
serviceInterface Id	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the service InterfaceId in the SOME/IP header matches.

Table A.19: SomeipSdRule

Class	**StateDependentFirewall**			
Note	Firewall rules that are defined in a firewall state **Tags:** atp.recommendedPackage=StateDependentFirewallRules			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
defaultAction	FirewallActionEnum	0..1	attr	This attribute defines a defaultAction in case that the VehicleMode is not yet set.
firewallRule Props (ordered)	FirewallRuleProps	*	aggr	Collection of firewall rules that apply in the vehicle mode
state	ModeDeclaration	*	iref	Reference to firewall states in which the Firewall is active. If one of the referenced ModeDeclarations is the current firewall state then the firewall rule shall be considered as active. **Tags:** atp.Splitkey=state, state.contextModeDeclaration GroupPrototype, state.targetModeDeclaration **InstanceRef implemented by:** FirewallStateInFirewall StateSwitchInterfaceInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.20: StateDependentFirewall

Class	**TcpRule**			
Note	Configuration of TCP filter rules.			
Base	*ARObject*, *TransportLayerRule*			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
numberOf ParallelTcp Sessions	PositiveInteger	0..1	attr	This attribute defines the maximal number of TCP Sessions that are allowed to be established.

▽

△

Class	**TcpRule**			
state Management BasedOnTcp Flags	Boolean	0..1	attr	This attribute defines whether the StateManagement is based on TCP flags or not.
timeoutCheck	PositiveInteger	0..1	attr	This attribute defines the TCP Session timeout in seconds

Table A.21: TcpRule

Class	***TransportLayerRule*** (abstract)			
Note	Configuration of filter rules on Transport Layer level.			
Base	*ARObject*			
Subclasses	TcpRule, UdpRule			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksum Verification	Boolean	0..1	attr	Defines whether checksum verification is performed or not.
maxDestination PortNumber	PositiveInteger	0..1	attr	Filter to match packets with the maximum destination UDP/TCP port number.
maxSourcePort Number	PositiveInteger	0..1	attr	Filter to match packets with the maximum source UDP/ TCP port number.
minDestination PortNumber	PositiveInteger	0..1	attr	Filter to match packets with the minimum destination UDP/TCP port number.
minSourcePort Number	PositiveInteger	0..1	attr	Filter to match packets with the minimum source UDP/ TCP port number.

Table A.22: TransportLayerRule

B Demands and constraints on Base Software (normative)

There are currently no demands or constraints on base software.

C Platform Extension Interfaces (normative)

This functional cluster does not specify any Platform Extension Interface.

D Not implemented requirements

There are no functional requirements that are not implemented or violated by this specification.

E Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

E.1 Traceable item history of this document according to AUTOSAR Release R25-11

E.1.1 Added Specification Items in R26-11

none

E.1.2 Changed Specification Items in R26-11

none

E.1.3 Deleted Specification Items in R26-11

none

E.1.4 Added Constraints in R26-11

none

E.1.5 Changed Constraints in R26-11

none

E.1.6 Deleted Constraints in R26-11

none

E.2 Traceable item history of this document according to AUTOSAR Release R25-11

E.2.1 Added Specification Items in R25-11

[AP_SWS_Fw_40013] [AP_SWS_Fw_60032] [AP_SWS_Fw_60034] [AP_SWS_Fw_60035]

E.2.2 Changed Specification Items in R25-11

[AP_SWS_Fw_60001] [AP_SWS_Fw_60002] [AP_SWS_Fw_60003] [AP_SWS_Fw_60020] [AP_SWS_Fw_60021] [AP_SWS_Fw_60022] [AP_SWS_Fw_60023] [AP_SWS_Fw_60024] [AP_SWS_Fw_60025] [AP_SWS_Fw_60026] [AP_SWS_Fw_60027] [AP_SWS_Fw_60028] [AP_SWS_Fw_60029] [AP_SWS_Fw_60030] [AP_SWS_Fw_60031] [AP_SWS_Fw_61000] [AP_SWS_Fw_82001] [AP_SWS_Fw_82006] [AP_SWS_Fw_83002] [AP_SWS_Fw_83003] [AP_SWS_Fw_83005] [AP_SWS_Fw_83007] [AP_SWS_Fw_83008] [AP_SWS_Fw_83009] [AP_SWS_Fw_83010] [AP_SWS_Fw_83011]

E.2.3 Deleted Specification Items in R25-11

none

E.2.4 Added Constraints in R25-11

none

E.2.5 Changed Constraints in R25-11

none

E.2.6 Deleted Constraints in R25-11

none

E.3 Traceable item history of this document according to AUTOSAR Release R24-11

E.3.1 Added Specification Items in R24-11

[AP_SWS_Fw_61000] [AP_SWS_Fw_83001] [AP_SWS_Fw_83002] [AP_SWS_Fw_83003] [AP_SWS_Fw_83004] [AP_SWS_Fw_83005] [AP_SWS_Fw_83006] [AP_SWS_Fw_83007] [AP_SWS_Fw_83008] [AP_SWS_Fw_83009] [AP_SWS_Fw_83010] [AP_SWS_Fw_83011] [AP_SWS_Fw_83012]

E.3.2 Changed Specification Items in R24-11

[AP_SWS_Fw_60001] [AP_SWS_Fw_60002] [AP_SWS_Fw_60003] [AP_SWS_Fw_60020] [AP_SWS_Fw_60021] [AP_SWS_Fw_60022] [AP_SWS_Fw_60023] [AP_SWS_Fw_60024] [AP_SWS_Fw_60025] [AP_SWS_Fw_60026] [AP_SWS_Fw_60027] [AP_SWS_Fw_60028] [AP_SWS_Fw_60029] [AP_SWS_Fw_60030] [AP_SWS_Fw_60031] [AP_SWS_Fw_80001] [AP_SWS_Fw_81001] [AP_SWS_Fw_81002] [AP_SWS_Fw_82002] [AP_SWS_Fw_82003] [AP_SWS_Fw_82004] [AP_SWS_Fw_82005] [AP_SWS_Fw_82006] [AP_SWS_Fw_82007] [AP_SWS_Fw_82008]

E.3.3 Deleted Specification Items in R24-11

none

E.3.4 Added Constraints in R24-11

[AP_SWS_Fw_CONSTR_00001]

E.3.5 Changed Constraints in R24-11

none

E.3.6 Deleted Constraints in R24-11

none

E.4 Traceable item history of this document according to AUTOSAR Release R23-11

E.4.1 Added Specification Items in R23-11

[AP_SWS_Fw_31001] [AP_SWS_Fw_31002] [AP_SWS_Fw_61000]

E.4.2 Changed Specification Items in R23-11

[AP_SWS_Fw_00001] [AP_SWS_Fw_00002] [AP_SWS_Fw_30001] [AP_SWS_Fw_30002] [AP_SWS_Fw_30003] [AP_SWS_Fw_30004] [AP_SWS_Fw_30005] [AP_SWS_Fw_30006] [AP_SWS_Fw_30007] [AP_SWS_Fw_30008] [AP_SWS_Fw_30009] [AP_SWS_Fw_30010] [AP_SWS_Fw_30011] [AP_SWS_Fw_30012] [AP_SWS_Fw_30013] [AP_SWS_Fw_30014] [AP_SWS_Fw_30015] [AP_SWS_Fw_30016] [AP_SWS_Fw_30017] [AP_SWS_Fw_30018] [AP_SWS_Fw_30019] [AP_SWS_Fw_30020] [AP_SWS_Fw_30021] [AP_SWS_Fw_30022] [AP_SWS_Fw_30023] [AP_SWS_Fw_30024] [AP_SWS_Fw_30025] [AP_SWS_Fw_30026] [AP_SWS_Fw_40001] [AP_SWS_Fw_40002] [AP_SWS_Fw_40003] [AP_SWS_Fw_40004] [AP_SWS_Fw_40005] [AP_SWS_Fw_40010] [AP_SWS_Fw_40012] [AP_SWS_Fw_60001] [AP_SWS_Fw_60002] [AP_SWS_Fw_60003] [AP_SWS_Fw_60004] [AP_SWS_Fw_60005] [AP_SWS_Fw_60006] [AP_SWS_Fw_60007] [AP_SWS_Fw_60008] [AP_SWS_Fw_60009] [AP_SWS_Fw_60010] [AP_SWS_Fw_60011] [AP_SWS_Fw_60012] [AP_SWS_Fw_60013] [AP_SWS_Fw_60014] [AP_SWS_Fw_60015] [AP_SWS_Fw_60016] [AP_SWS_Fw_60017] [AP_SWS_Fw_60018] [AP_SWS_Fw_60019] [AP_SWS_Fw_60020] [AP_SWS_Fw_60021] [AP_SWS_Fw_60022] [AP_SWS_Fw_60023] [AP_SWS_Fw_60024] [AP_SWS_Fw_60025] [AP_SWS_Fw_60026] [AP_SWS_Fw_60027] [AP_SWS_Fw_60028] [AP_SWS_Fw_60029] [AP_SWS_Fw_60030] [AP_SWS_Fw_60031] [AP_SWS_Fw_80001] [AP_SWS_Fw_81001] [AP_SWS_Fw_81002] [AP_SWS_Fw_82001] [AP_SWS_Fw_82002] [AP_SWS_Fw_82003] [AP_SWS_Fw_82004] [AP_SWS_Fw_82005] [AP_SWS_Fw_82006] [AP_SWS_Fw_82007] [AP_SWS_Fw_82008]

E.4.3 Deleted Specification Items in R23-11

none

E.5 Traceable item history of this document according to AUTOSAR Release R22-11

E.5.1 Added Specification Items in R22-11

[AP_SWS_Fw_00001] [AP_SWS_Fw_00002] [AP_SWS_Fw_30001] [AP_SWS_Fw_30002] [AP_SWS_Fw_30003] [AP_SWS_Fw_30004] [AP_SWS_Fw_30005]

[AP_SWS_Fw_30006] [AP_SWS_Fw_30007] [AP_SWS_Fw_30008] [AP_SWS_Fw_30009] [AP_SWS_Fw_30010] [AP_SWS_Fw_30011] [AP_SWS_Fw_30012] [AP_SWS_Fw_30013] [AP_SWS_Fw_30014] [AP_SWS_Fw_30015] [AP_SWS_Fw_30016] [AP_SWS_Fw_30017] [AP_SWS_Fw_30018] [AP_SWS_Fw_30019] [AP_SWS_Fw_30020] [AP_SWS_Fw_30021] [AP_SWS_Fw_30022] [AP_SWS_Fw_30023] [AP_SWS_Fw_30024] [AP_SWS_Fw_30025] [AP_SWS_Fw_30026] [AP_SWS_Fw_40001] [AP_SWS_Fw_40002] [AP_SWS_Fw_40003] [AP_SWS_Fw_40004] [AP_SWS_Fw_40005] [AP_SWS_Fw_40006] [AP_SWS_Fw_40007] [AP_SWS_Fw_40008] [AP_SWS_Fw_40009] [AP_SWS_Fw_40010] [AP_SWS_Fw_40011] [AP_SWS_Fw_40012] [AP_SWS_Fw_60001] [AP_SWS_Fw_60002] [AP_SWS_Fw_60003] [AP_SWS_Fw_60004] [AP_SWS_Fw_60005] [AP_SWS_Fw_60006] [AP_SWS_Fw_60007] [AP_SWS_Fw_60008] [AP_SWS_Fw_60009] [AP_SWS_Fw_60010] [AP_SWS_Fw_60011] [AP_SWS_Fw_60012] [AP_SWS_Fw_60013] [AP_SWS_Fw_60014] [AP_SWS_Fw_60015] [AP_SWS_Fw_60016] [AP_SWS_Fw_60017] [AP_SWS_Fw_60018] [AP_SWS_Fw_60019] [AP_SWS_Fw_60020] [AP_SWS_Fw_60021] [AP_SWS_Fw_60022] [AP_SWS_Fw_60023] [AP_SWS_Fw_60024] [AP_SWS_Fw_60025] [AP_SWS_Fw_60026] [AP_SWS_Fw_60027] [AP_SWS_Fw_60028] [AP_SWS_Fw_60029] [AP_SWS_Fw_60030] [AP_SWS_Fw_60031] [AP_SWS_Fw_80001] [AP_SWS_Fw_81001] [AP_SWS_Fw_81002] [AP_SWS_Fw_82001] [AP_SWS_Fw_82002] [AP_SWS_Fw_82003] [AP_SWS_Fw_82004] [AP_SWS_Fw_82005] [AP_SWS_Fw_82006] [AP_SWS_Fw_82007] [AP_SWS_Fw_82008]

E.5.2 Changed Specification Items in R22-11

none

E.5.3 Deleted Specification Items in R22-11

none