

Document Title	Specification of Firewall for Adaptive Platform
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1063

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R23-11

Document Change History			
Date	Release	Changed by	Description
2023-11-23	R23-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Minor bugfixes
2022-11-24	R22-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • 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.

Contents

1	Introduction and functional overview	5
2	Acronyms and Abbreviations	6
2.1	Acronyms	6
2.2	Abbreviations	6
3	Related documentation	7
3.1	Input documents & related standards and norms	7
3.2	Further applicable specification	7
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	10
6	Requirements Tracing	11
7	Functional specification	12
7.1	Architecture Overview	12
7.2	Functional cluster life-cycle	14
7.3	Network packet inspection	14
7.3.1	Stateless packet inspection	15
7.3.1.1	Inspection of not modeled protocols	16
7.3.2	Stateful packet inspection	17
7.3.3	Deep packet inspection	17
7.3.3.1	SOME/IP	18
7.3.3.2	DDS	19
7.3.3.3	DoIP	20
7.3.3.4	Generic inspection	20
7.4	Network packet filtering	21
7.4.1	Allowlists and Blocklists	21
7.4.2	Rate limiting	22
7.4.3	State dependent filtering	22
7.5	Firewall Rule Management	23
7.6	Security Events	24
7.6.1	SEvs raised by the firewall	24
7.6.2	Raising SEvs	30
8	API specification	34
8.1	API Header Files	34
8.2	API Common Data Types	34
8.3	API Reference	35
8.3.1	FirewallStateSwitchInterface	35

8.3.1.1	SwitchFirewallState	37
9	Service Interfaces	39
A	Mentioned Manifest Elements	40
B	Platform Extension API (normative)	52
C	Interfaces to other Functional Clusters (informative)	53
C.1	Overview	53
C.2	Interface Tables	53
D	History of Constraints and Specification Items	54
D.1	Traceable item history of this document according to AUTOSAR Release R22-11	54
D.1.1	Added Specification Items in R22-11	54
D.1.2	Changed Specification Items in R22-11	58
D.1.3	Deleted Specification Items in R22-11	58
D.2	Constraint and Specification Item History of this document according to AUTOSAR Release 23-11	58
D.2.1	Added Specification Items in R23-11	58
D.2.2	Changed Specification Items in R23-11	58
D.2.3	Deleted Specification Items in R23-11	59

1 Introduction and functional overview

This specification describes the functionality and services for the functional cluster 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 relevant to the FC Firewall module that are not included 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 Rules 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-RTSPS	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:

- **RS_Firewall** [2]: Requirement specification of the AUTOSAR firewall on Foundation level.
- **TPS_ManifestSpecification** [3]: 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] Requirements on Firewall
AUTOSAR_FO_RS_Firewall
- [3] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification
- [4] Specification of Adaptive Platform Core
AUTOSAR_AP_SWS_Core
- [5] Explanation of Adaptive Platform Software Architecture
AUTOSAR_AP_EXP_SWArchitecture
- [6] IEEE Standard for Ethernet
<https://ieeexplore.ieee.org/document/7428776>
- [7] SOME/IP Protocol Specification
AUTOSAR_FO_PRS_SOMEIPProtocol
- [8] SOME/IP Service Discovery Protocol Specification
AUTOSAR_FO_PRS_SOMEIPServiceDiscoveryProtocol
- [9] DDS Interoperability Wire Protocol, Version 2.2
<http://www.omg.org/spec/DDSI-RTPS/2.2>
- [10] Road vehicles – Diagnostic communication over Internet Protocol (DoIP) – Part 2: Network and transport layer requirements and services (Release 2019-12)
<https://www.iso.org>

3.2 Further applicable specification

AUTOSAR provides a core specification [4] which is also applicable for the [FC Firewall](#). The chapter "General requirements for all FunctionalClusters" of this specifica-

tion shall be considered as an additional and required specification for implementation of the [FC Firewall](#).

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

This chapter provides an overview of the dependencies to other Functional Clusters in the AUTOSAR Adaptive Platform. Section 5.1 “Provided Interfaces” lists the interfaces provided by `Firewall` to other Functional Clusters. Section 5.2 “Required Interfaces” lists the interfaces required by `Firewall`.

A detailed technical architecture documentation of the AUTOSAR Adaptive Platform is provided in [5].

5.1 Provided Interfaces

Interface	Functional Cluster	Purpose
No provided interfaces		

Table 5.1: Interfaces provided to other Functional Clusters

5.2 Required Interfaces

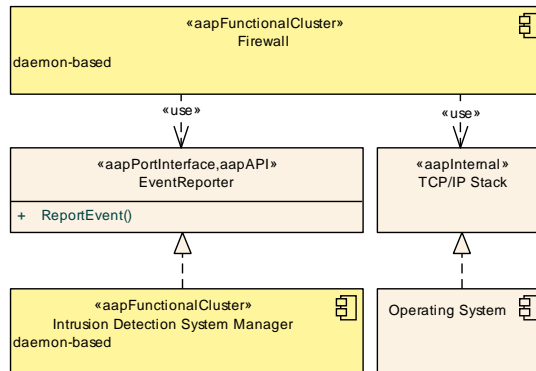


Figure 5.1: Interfaces required by Firewall from other Functional Clusters

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

Functional Cluster	Interface	Purpose
Intrusion Detection System Manager	EventReporter	The <code>Firewall</code> uses this interface to report standardized security events.

Table 5.2: Interfaces required from other Functional Clusters

6 Requirements Tracing

The following tables reference the requirements specified in [2] 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
[FO_RS_Fw_00001]	Stateless filtering of network traffic	[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]
[FO_RS_Fw_00002]	Stateful filtering of network traffic	[AP_SWS_Fw_30012] [AP_SWS_Fw_30013] [AP_SWS_Fw_30014]
[FO_RS_Fw_00003]	Deep Packet Inspection of network traffic	[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]
[FO_RS_Fw_00004]	Allow list and block list configuration	[AP_SWS_Fw_40001] [AP_SWS_Fw_40002] [AP_SWS_Fw_40003]
[FO_RS_Fw_00005]	Rule-Based filtering of network traffic	[AP_SWS_Fw_30001] [AP_SWS_Fw_30002] [AP_SWS_Fw_31001] [AP_SWS_Fw_31002]
[FO_RS_Fw_00006]	Rate Limiting	[AP_SWS_Fw_40004] [AP_SWS_Fw_40005]
[FO_RS_Fw_00007]	State-dependent Filtering	[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_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]
[FO_RS_Fw_00008]	Raising of security Alerts	[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_61000]
[FO_RS_Fw_00010]	Initialization of the Firewall	[AP_SWS_Fw_00001] [AP_SWS_Fw_00002]

Table 6.1: RequirementsTracing

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.

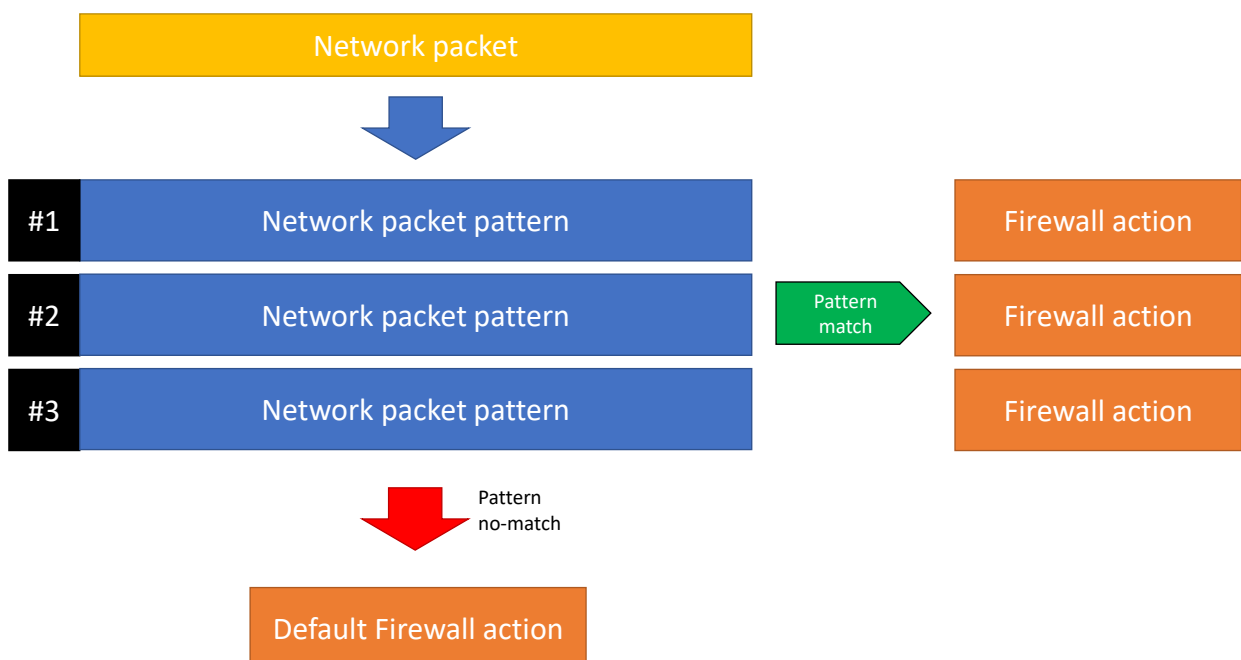


Figure 7.1: Pattern matching mechanism

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 7.4.3. 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

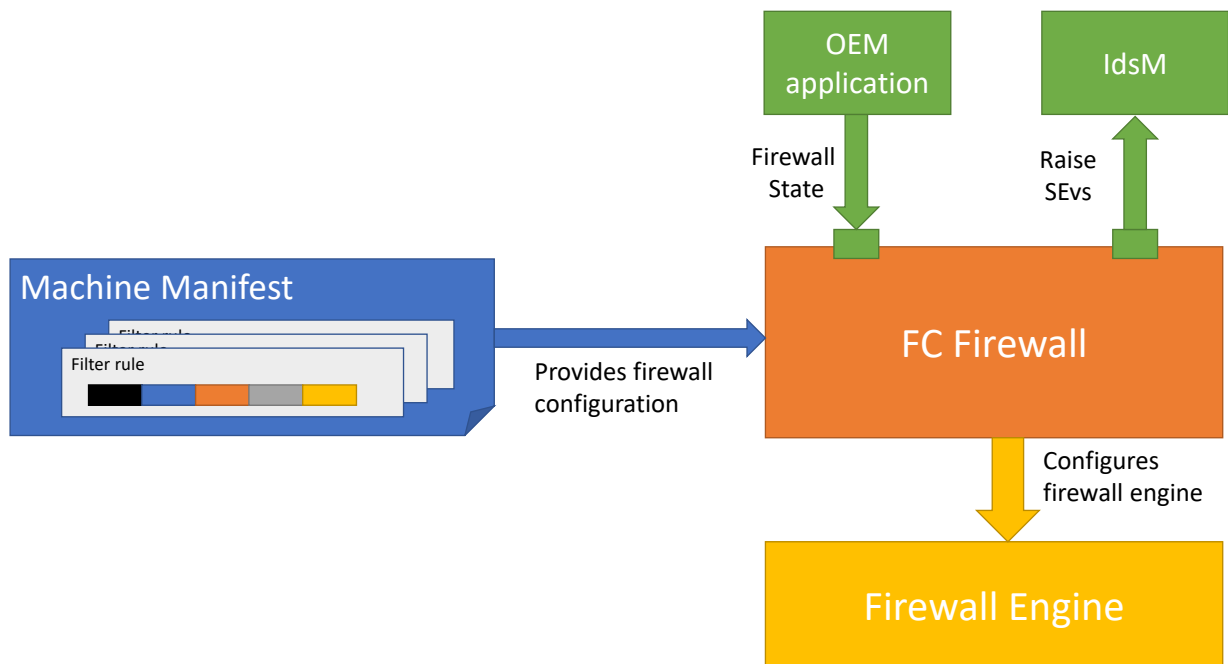


Figure 7.2: Architecture of the FC Firewall

This chapter is structured as follows:

- Sec. 7.2 describes the lifecycle of the FC Firewall
- Sec. 7.3 describes the network packet inspection, i.e. the pattern-matching part of the `FirewallRules`
- Sec. 7.4 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. 7.5 describes the management of Firewall rules, i.e., how to add/remove/change rules and (de-)activate rules during runtime
- Sec. 7.6 describes the security events raised by the Firewall

7.2 Functional cluster life-cycle

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

[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.](*FO_RS_Fw_00010*)

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.

[AP_SWS_Fw_00002] [When `ara::core::Deinitialize` is called, the `FC Firewall` shall close all acquired handles and free all access structures.](*FO_RS_Fw_00010*)

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

7.3 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.](*FO_RS_Fw_00005*)

The possible actions in case of a pattern match are described in Sec. 7.4.

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.] ([FO_RS_Fw_00005](#))

[AP_SWS_Fw_31002] [For egress traffic, only [FirewallRules](#) that are referenced by [FirewallRuleProps.matchingEgressRule](#) shall be considered for network traffic inspection.] ([FO_RS_Fw_00005](#))

[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.] ([FO_RS_Fw_00005](#))

7.3.1 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] ([FO_RS_Fw_00001](#))

[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] ([FO_RS_Fw_00001](#))

[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] ([FO_RS_Fw_00001](#))

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`](*FO_RS_Fw_00001*)

[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`](*FO_RS_Fw_00001*)

[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`](*FO_RS_Fw_00001*)

[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`](*FO_RS_Fw_00001*)

[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`](*FO_RS_Fw_00001*)

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.](*FO_RS_Fw_00001*)

7.3.1.1 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 [6]). 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.

<i>EtherType</i>	<i>Protocol</i>
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. [7.3.3.4](#) and allows for detailed inspection of protocols that are not modeled within the [FC Firewall](#) as well as inspection of payload data.

7.3.2 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.] ([FO_RS_Fw_00002](#))

[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.] ([FO_RS_Fw_00002](#))

[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.] ([FO_RS_Fw_00002](#))

7.3.3 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

7.3.3.1 SOME/IP

For [SOME/IP](#) [7] 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.] ([FO_RS_Fw_00003](#))

[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.] ([FO_RS_Fw_00003](#))

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] ([FO_RS_Fw_00003](#))

The [FC Firewall](#) also supports inspection of the [SOME/IP](#) service discovery protocol [8]. 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.]([FO_RS_Fw_00003](#))

[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.]([FO_RS_Fw_00003](#))

[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](#) [8]. If all values match, the [SomeipSdRule](#) is considered a match. Otherwise the [SomeipSdRule](#) is considered a no-match]([FO_RS_Fw_00003](#))

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](#)]([FO_RS_Fw_00003](#))

7.3.3.2 DDS

Deep packet inspection of DDS messages is based on the DDS Interoperability Wire Protocol ([DDS-RTPS](#) [9]), 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.]([FO_RS_Fw_00003](#))

[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.](*FO_RS_Fw_00003*)

7.3.3.3 DoIP

The `FC Firewall` supports deep packet inspection of `DoIP` messages [10], 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. 7.3.3.4.

[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](*FO_RS_Fw_00003*)

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`](*FO_RS_Fw_00003*)

7.3.3.4 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.](*FO_RS_Fw_00003*)

7.4 Network packet filtering

After describing the rule-based network packet inspection process based on pattern-matching in chapter 7.3, this chapter specifies the associated filtering mechanisms supported by the `FC Firewall`. Section 7.4.1 describes the pattern-matching-based filtering approach using `Allowlists` and `Blocklists`, section 7.4.2 specifies the rate limiting feature of the `FC Firewall` and section 7.4.3 outlines the state-dependent filtering mechanism based on configurable `Firewall States`.

7.4.1 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] (*FO_RS_Fw_00004*)

[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] (*FO_RS_Fw_00004*)

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.] (*FO_RS_Fw_00004*)

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 for more details.

7.4.2 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] (*FO_RS_Fw_00006*)

[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.] (*FO_RS_Fw_00006*)

7.4.3 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]{DRAFT} [The `FC Firewall` shall ensure that only one `StateDependentFirewall` is active] (*FO_RS_Fw_00007*)

[AP_SWS_Fw_40007]{DRAFT} [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] (*FO_RS_Fw_00007*)

[AP_SWS_Fw_40008]{DRAFT} [For no-match cases, the `StateDependentFirewall.defaultAction` defined in the currently active `StateDependentFirewall` shall be used] (*FO_RS_Fw_00007*)

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

[AP_SWS_Fw_40009]{DRAFT} [If a `ModeDeclaration` is reported to the `FC Firewall` by means of `ara::fw::FirewallStateSwitchInterface::SwitchFirewallState`, the referenced `StateDependentFirewall` shall be considered as active.](*FO_RS_Fw_00007*)

[AP_SWS_Fw_40010]{DRAFT} [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`.](*FO_RS_Fw_00007*)

[AP_SWS_Fw_40011]{DRAFT} [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`.](*FO_RS_Fw_00007*)

[AP_SWS_Fw_40012]{DRAFT} [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`.](*FO_RS_Fw_00007*)

7.5 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.

7.6 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 ([SEvs](#)) to the [IdsM](#). To this end, the [FC Firewall](#) specifies a set of [SEvs](#) (see [Sec. 7.6.1](#)) as well as conditions on when to raise them (see [Sec. 7.6.2](#)).

7.6.1 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_DATAINKLAYER_MISMATCH	A network packet was blocked due to a rule mismatch on data link layer.	77
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





Name	Description	ID
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

](FO_RS_Fw_00008)

[AP_SWS_Fw_60001] [

SEV component	Description
Name	SEV_FW_PACKET_BLOCKED_DATA_LINK_LAYER_MISMATCH
Description	A network packet was blocked due to a rule mismatch on data link layer
SEV ID	77
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete Ethernet header

Table 7.2: Data link layer SEV

](FO_RS_Fw_00008)

[AP_SWS_Fw_60020] [

SEV component	Description
Name	SEV_FW_PACKET_BLOCKED_IPV4_MISMATCH
Description	A network packet was blocked due to a rule mismatch on IPv4 layer
SEV ID	51
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete IPv4 header

Table 7.3: IPv4 SEV

](FO_RS_Fw_00008)

[AP_SWS_Fw_60021] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_IPV6_MISMATCH
Description	A network packet was blocked due to a rule mismatch on IPv6 layer
SEV ID	52
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete IPv6 header

Table 7.4: IPv6 SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60022] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_ICMP_MISMATCH
Description	A network packet was blocked due to a rule mismatch within the ICMP protocol
SEV ID	53
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete ICMP header (type, code, checksum)

Table 7.5: ICMP SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60023] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_TCP_MISMATCH
Description	A network packet was blocked due to a rule mismatch on TCP layer
SEV ID	54
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete TCP header

Table 7.6: TCP SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60024] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_UDP_MISMATCH
Description	A network packet was blocked due to a rule mismatch on UDP layer
SEV ID	55
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete UDP header

Table 7.7: UDP SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60025] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_SOMEIP_MISMATCH
Description	A network packet was blocked due to a rule mismatch in the SOME/IP protocol
SEV ID	56
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete SOME/IP header

Table 7.8: SOME/IP SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60026] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_SOMEIPSD_MISMATCH
Description	A network packet was blocked due to a rule mismatch in the SOME/IP SD protocol
SEV ID	57
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete SOME/IP SD header

Table 7.9: SOME/IP SD SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60027] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_DDS_MISMATCH
Description	A network packet was blocked due to a rule mismatch in the DDS-RTPS protocol
SEV ID	58
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete DDS-RTPS Header

Table 7.10: DDS SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60028] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_DOIP_MISMATCH
Description	A network packet was blocked due to a rule mismatch in the DoIP protocol
SEV ID	59
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete DoIP header

Table 7.11: DoIP SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60029] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_GENERIC_MISMATCH
Description	A network packet was blocked due to a rule mismatch on generic inspection level
SEV ID	60
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname

Table 7.12: Generic SEV

]([FO_RS_Fw_00008](#))

Additionally, the [FC Firewall](#) also specifies a set of [SEVs](#) that are focusing on the stateful properties of TCP connections:

[AP_SWS_Fw_60002] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_TCP_MAXCONNECTIONS
Description	A network packet was blocked due to the maximal number of open TCP connections was reached
SEV ID	61
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete TCP Header

Table 7.13: TCP Max Connections SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60030] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_TCP_TIMEOUT
Description	A network packet was blocked due to TCP timeout
SEV ID	62
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Complete TCP header

Table 7.14: TCP Timeout SEV

]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60031] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_TCP_STATETRANSITION
Description	A network packet was blocked due to an invalid TCP state transition
SEV ID	63
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname

Table 7.15: TCP state machine SEV

]([FO_RS_Fw_00008](#))

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

[AP_SWS_Fw_60003] [

<i>SEV component</i>	<i>Description</i>
Name	SEV_FW_PACKET_BLOCKED_RATELIMIT
Description	A network packet was blocked due to the rate limit was reached
SEV ID	64
Context Data	<ul style="list-style-type: none"> • FirewallRule Shortname • Source MAC address

Table 7.16: Rate limit SEV

](FO_RS_Fw_00008)

7.6.2 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.](FO_RS_Fw_00008)

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 Layers 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 `FirewallRules` is Ethernet, the `FC Firewall` shall raise the `SEv SEV_FW_PACKET_BLOCKED_DATALINKLAYER_MISMATCH` to the `IdsM.`]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60006] [If a network packet is blocked by the default action and the network protocol that was not matching the `FirewallRules` is IPv4, the `FC Firewall` shall raise the `SEv SEV_FW_PACKET_BLOCKED_IPV4_MISMATCH` to the `IdsM.`]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60007] [If a network packet is blocked by the default action and the network protocol that was not matching the `FirewallRules` is IPv6, the `FC Firewall` shall raise the `SEv SEV_FW_PACKET_BLOCKED_IPV6_MISMATCH` to the `IdsM.`]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60008] [If a network packet is blocked by the default action and the network protocol that was not matching the `FirewallRules` is ICMP, the `FC Firewall` shall raise the `SEv SEV_FW_PACKET_BLOCKED_ICMP_MISMATCH` to the `IdsM.`]([FO_RS_Fw_00008](#))

[AP_SWS_Fw_60009] [If a network packet is blocked by the default action and the network protocol that was not matching the `FirewallRules` is TCP, the `FC Firewall` shall raise the `SEv SEV_FW_PACKET_BLOCKED_TCP_MISMATCH` to the `IdsM.`]([FO_RS_Fw_00008](#))

[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.`] (*FO_RS_Fw_00008*)

[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.`] (*FO_RS_Fw_00008*)

[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.`] (*FO_RS_Fw_00008*)

[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.`] (*FO_RS_Fw_00008*)

[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.`] (*FO_RS_Fw_00008*)

[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.`] (*FO_RS_Fw_00008*)

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.`] (*FO_RS_Fw_00008*)

[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.`] (*FO_RS_Fw_00008*)

[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.`] (*FO_RS_Fw_00008*)

Finally, network packets can also be dropped due to the rate limiting feature described in Sec. [7.4.2](#)

[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*.] ([FO_RS_Fw_00008](#))

8 API specification

8.1 API Header Files

[AP_SWS_Fw_80001] Generated header files for `FirewallStateSwitchInterface` [The `FC Firewall` shall provide a header file for each `FirewallStateSwitchInterface` by using the file name `<name>.h`, where `<name>` is the `FirewallStateSwitchInterface.shortName`. This header file shall be provided in the folder: `ara/fw/states.`] (*FO_RS_Fw_00007*)

8.2 API Common Data Types

This chapter describes the standardized types provided by the `ara::fw` API. The `ara::fw` API is based on the `ara::core` types defined in [4].

The `FC Firewall` offers the possibility to switch between user-defined `Firewall States`. The types used by these states are generated based on the input generation.

An Enumeration is not a plain primitive data type, but a structural description defined with a set of custom identifiers known as *enumerators* representing the possible values. In C++, an enumeration is a first-class object and can take any of these enumerators as a value.

For each `FirewallStateSwitchInterface`, an enumeration is generated containing the corresponding `Firewall States`.

[AP_SWS_Fw_81001] Enumeration for `FirewallStateSwitchInterface` [For each `FirewallStateSwitchInterface`, there shall exist the corresponding type declaration as:

```
enum class FirewallStateSwitchInterface.shortName : std::uint32_t {
    <enumerator-list>
};
```

where `<enumerator-list>` are the enumerators as defined by [AP_SWS_Fw_81002]. It shall be provided in the namespace `ara::fw::states.`] (*FO_RS_Fw_00007*)

[AP_SWS_Fw_81002] Definition of enumerators of `FirewallStateSwitchInterface` [For each `ModeDeclaration` contained in the `FirewallStateSwitchInterface`, there shall exist the corresponding enumeration nested in the declaration defined by [AP_SWS_Fw_81001] as:

```
<enumeratorLiteral> = <initializer><suffix>,
```

where:

`<enumeratorLiteral>` is the `ModeDeclaration.shortName`

<initializer> is the `ModeDeclaration.value`

<suffix> shall be "U".

]([FO_RS_Fw_00007](#))

For example, this can generate:

```
enum class FirewallStates : std::uint32_t
{
    DefaultState = 0U,
    DrivingState = 1U,
    DiagnosticState = 2U
};
```

8.3 API Reference

8.3.1 FirewallStateSwitchInterface

The `Firewall States` can be switched by using the `FirewallStateSwitchInterface`

[AP_SWS_Fw_82001]{DRAFT} Definition of API class `ara::fw::FirewallStateSwitchInterface` [

Kind:	class	
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 {...};	
Template param:	typename EnumT	An enum type that contains a list of firewall states
Description:	FirewallStateSwitchInterface Class.	

]([FO_RS_Fw_00007](#))

[AP_SWS_Fw_82002]{DRAFT} Definition of API function `ara::fw::FirewallStateSwitchInterface::FirewallStateSwitchInterface` [

Kind:	function	
Header file:	#include "ara/fw/firewall_state.h"	
Scope:	class ara::fw::FirewallStateSwitchInterface	
Symbol:	FirewallStateSwitchInterface(const ara::core::InstanceSpecifier &instance)	
Syntax:	explicit FirewallStateSwitchInterface (const ara::core::InstanceSpecifier &instance);	
Parameters (in):	instance	Instance specifier of the Port typed with FirewallStateSwitchInterface.



△

Exception Safety:	not exception safe
Description:	Creation of a FirewallStateSwitchInterface.

](FO_RS_Fw_00007)

[AP_SWS_Fw_82003]{DRAFT} Definition of API function ara::fw::FirewallStateSwitchInterface::~~FirewallStateSwitchInterface [

Kind:	function
Header file:	#include "ara/fw/firewall_state.h"
Scope:	class ara::fw::FirewallStateSwitchInterface
Symbol:	~FirewallStateSwitchInterface()
Syntax:	~FirewallStateSwitchInterface () noexcept;
Exception Safety:	noexcept
Description:	Destructor of a FirewallStateSwitchInterface.

](FO_RS_Fw_00007)

[AP_SWS_Fw_82004]{DRAFT} Definition of API function ara::fw::FirewallStateSwitchInterface::FirewallStateSwitchInterface [

Kind:	function
Header file:	#include "ara/fw/firewall_state.h"
Scope:	class ara::fw::FirewallStateSwitchInterface
Symbol:	FirewallStateSwitchInterface(const FirewallStateSwitchInterface &se)
Syntax:	FirewallStateSwitchInterface (const FirewallStateSwitchInterface &se)=delete;
Description:	The copy constructor for FirewallStateSwitchInterface shall not be used.

](FO_RS_Fw_00007)

[AP_SWS_Fw_82005]{DRAFT} Definition of API function ara::fw::FirewallStateSwitchInterface::operator= [

Kind:	function
Header file:	#include "ara/fw/firewall_state.h"
Scope:	class ara::fw::FirewallStateSwitchInterface
Symbol:	operator=(const FirewallStateSwitchInterface &se)
Syntax:	FirewallStateSwitchInterface & operator= (const FirewallStateSwitchInterface &se)=delete;
Description:	The copy assignment operator for FirewallStateSwitchInterface shall not be used.

](FO_RS_Fw_00007)

[AP_SWS_Fw_82006]{DRAFT} Definition of API function ara::fw::FirewallStateSwitchInterface::FirewallStateSwitchInterface [

Kind:	function	
Header file:	#include "ara/fw/firewall_state.h"	
Scope:	class ara::fw::FirewallStateSwitchInterface	
Symbol:	FirewallStateSwitchInterface(FirewallStateSwitchInterface &&se)	
Syntax:	FirewallStateSwitchInterface (FirewallStateSwitchInterface &&se) noexcept;	
Parameters (in):	se	The FirewallStateSwitchInterface object to be moved.
Exception Safety:	noexcept	
Description:	Move constructor for FirewallStateSwitchInterface.	

]([FO_RS_Fw_00007](#))

[AP_SWS_Fw_82007]{DRAFT} Definition of API function ara::fw::FirewallStateSwitchInterface::operator= [

Kind:	function	
Header file:	#include "ara/fw/firewall_state.h"	
Scope:	class ara::fw::FirewallStateSwitchInterface	
Symbol:	operator=(FirewallStateSwitchInterface &&se)	
Syntax:	FirewallStateSwitchInterface & operator= (FirewallStateSwitchInterface &&se) noexcept;	
Parameters (in):	se	The FirewallStateSwitchInterface object to be moved.
Return value:	FirewallStateSwitchInterface &	The moved FirewallStateSwitchInterface object.
Exception Safety:	noexcept	
Description:	Move assignment operator for FirewallStateSwitchInterface.	

]([FO_RS_Fw_00007](#))

8.3.1.1 SwitchFirewallState

[AP_SWS_Fw_82008]{DRAFT} Definition of API function ara::fw::FirewallStateSwitchInterface::SwitchFirewallState [

Kind:	function	
Header file:	#include "ara/fw/firewall_state.h"	
Scope:	class ara::fw::FirewallStateSwitchInterface	
Symbol:	SwitchFirewallState(EnumT firewallState)	
Syntax:	ara::core::Future< void > SwitchFirewallState (EnumT firewallState) noexcept;	
Parameters (in):	firewallState	The FirewallState to be set.
Return value:	ara::core::Future< void >	–
Exception Safety:	noexcept	





Errors:	FirewallErrorDomain::Fw Errc::kServiceNot Available	Communication to Firewall daemon is broken, i.e. state is not switched
	FirewallErrorDomain::Fw Errc::kInvalidState DependentFirewall	This firewallState is not used by any StateDependentFirewall rule-sets
Description:	This method sets the FirewallState for the FC Firewall.	

|(FO_RS_Fw_00007)

9 Service Interfaces

No content

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	DataLinkLayerRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules on the DataLink layer Tags: atp.Status=candidate			
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. Tags: atp.Status=candidate
destinationMac AddressMask	MacAddressString	0..1	attr	Filter to match packets with the destination MAC address range. The destinationMacAddress with the destinationMacAddressMask defines the MAC address range. Tags: atp.Status=candidate
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. Tags: atp.Status=candidate
sourceMac Address	MacAddressString	0..1	attr	Filter to match packets with the source MAC address. Tags: atp.Status=candidate
sourceMac AddressMask	MacAddressString	0..1	attr	Filter to match packets with the source MAC address range. The sourceMacAddress with the sourceMacAddressMask defines the MAC address range. Tags: atp.Status=candidate
vlanId	PositiveInteger	0..1	attr	Filter of packets with a specific VlanId. Tags: atp.Status=candidate
vlanPriority	PositiveInteger	0..1	attr	Filter of packets with a specific Vlan priority. Tags: atp.Status=candidate

Table A.1: DataLinkLayerRule

Class	DdsRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of a DDS firewall rule Tags: atp.Status=candidate			
Base	ARObject			
Aggregated by	FirewallRule.ddsRule			
Attribute	Type	Mult.	Kind	Note





Class	DdsRule			
appld	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the appld in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches. Tags: atp.Status=candidate
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. Tags: atp.Status=candidate
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. Tags: atp.Status=candidate
majorProtocolVersion	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the major ProtocolVersion in the DDSI-RTPS header matches. Tags: atp.Status=candidate
minorProtocolVersion	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the minor ProtocolVersion in the DDSI-RTPS header matches. Tags: atp.Status=candidate
productId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the productId in the DDSI-RTPS header matches. Tags: atp.Status=candidate
readerEntityId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the readerEntity ID in a DDSI-RTPS submessage matches Tags: atp.Status=candidate
submessageType	PositiveInteger	0..1	attr	Defines the allowed submessage type in the DDSI-RTPS message Tags: atp.Status=candidate
vendorId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the vendorId in the DDSI-RTPS header matches. Tags: atp.Status=candidate
writerEntityId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the writerEntity ID in a DDSI-RTPS submessage matches Tags: atp.Status=candidate

Table A.2: DdsRule

Class	DolpRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of a generic firewall rule Tags: atp.Status=candidate			
Base	ARObject			
Aggregated by	FirewallRule.dolpRule			
Attribute	Type	Mult.	Kind	Note
destinationMaxAddress	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the destination Address is smaller or equal than destinationMaxAddress. Tags: atp.Status=candidate





Class	DoIPRule			
destinationMinAddress	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the destination Address is greater or equal than destinationMinAddress. Tags: atp.Status=candidate
inverseProtocolVersion	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the inverseprotocolVersion in the DoIP header matches. Tags: atp.Status=candidate
payloadLength	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the payload Length in the DoIP header matches. Tags: atp.Status=candidate
payloadType	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the payloadType in the DoIP header matches. Tags: atp.Status=candidate
protocolVersion	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the protocol Version in the DoIP header matches. Tags: atp.Status=candidate
sourceMaxAddress	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the source Address is smaller or equal than sourceMaxAddress. Tags: atp.Status=candidate
sourceMinAddress	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the source Address is greater or equal than sourceMinAddress.. Tags: atp.Status=candidate
udsService	PositiveInteger	0..1	attr	Filter to match DoIP messages that contain the uds Service. Tags: atp.Status=candidate

Table A.3: DoIPRule

Class	FirewallRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Firewall Rule that defines the control information in individual packets. Tags: atp.Status=candidate atp.recommendedPackage=FirewallRules			
Base	<i>ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDesignElement, UploadablePackageElement</i>			
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). Tags: atp.Status=candidate
dataLinkLayerRule	DataLinkLayerRule	0..1	aggr	Configuration of rules on the Data Link Layer Tags: atp.Status=candidate
ddsRule	DdsRule	0..1	aggr	Configuration of firewall rules for DDS. Tags: atp.Status=candidate
doIPRule	DoIPRule	0..1	aggr	Configuration of firewall rules for DoIP messages Tags: atp.Status=candidate





Class	FirewallRule			
networkLayerRule	NetworkLayerRule	0..1	aggr	Configuration of rules on the Network Layer Tags: atp.Status=candidate
payloadBytePatternRule	PayloadBytePatternRule	*	aggr	Configuration of generic firewall rules Tags: atp.Status=candidate
refillAmount	PositiveInteger	0..1	attr	This attribute defines the output rate that describes how many packets leave the queue per second (leaky-bucket Algorithm). Tags: atp.Status=candidate
someipRule	SomeipProtocolRule	0..1	aggr	Configuration of firewall rules for SOME/IP messages Tags: atp.Status=candidate
someipSdRule	SomeipSdRule	0..1	aggr	Configuration of firewall rules for SOME/IP Service Discovery messages Tags: atp.Status=candidate
transportLayerRule	TransportLayerRule	0..1	aggr	Configuration of rules on the Transport Layer Tags: atp.Status=candidate

Table A.4: FirewallRule

Class	FirewallRuleProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Firewall rule that is defined by an action that is performed if the referenced pattern matches. Tags: atp.Status=candidate			
Base	<i>ARObject</i>			
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. Tags: atp.Status=candidate
matchingEgressRule (ordered)	FirewallRule	*	ref	This element defines an egress rule expression against which the network traffic is matched. Tags: atp.Status=candidate
matchingIngressRule (ordered)	FirewallRule	*	ref	This element defines an ingress rule expression against which the network traffic is matched. Tags: atp.Status=candidate

Table A.5: FirewallRuleProps

Class	FirewallStateSwitchInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Firewall mode. Tags: atp.Status=candidate atp.recommendedPackage=FirewallStateSwitchPortInterfaces			
Base	<i>ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable</i>			
Aggregated by	ARPackage.element			





Class				
FirewallStateSwitchInterface				
Attribute	Type	Mult.	Kind	Note
firewallStateMachine	ModeDeclarationGroup Prototype	*	aggr	The state machine of this firewall interface. Tags: atp.Status=candidate

Table A.6: FirewallStateSwitchInterface

Class				
IcmpRule				
Package				
M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall				
Note				
Configuration of filter rules for ICMP (Internet Control Message Protocol). Tags: atp.Status=candidate				
Base				
ARObject				
Aggregated by				
Ipv4Rule.icmpRule, Ipv6Rule.icmpRule				
Attribute	Type	Mult.	Kind	Note
checksumVerification	Boolean	0..1	attr	Defines whether a Icmp header checksum verification is performed or not. Tags: atp.Status=candidate
code	PositiveInteger	0..1	attr	Filter to match packets with the Icmp code. Tags: atp.Status=candidate
type	PositiveInteger	0..1	attr	Filter to match packets with the Icmp type. Tags: atp.Status=candidate

Table A.7: IcmpRule

Class				
Ipv4Rule				
Package				
M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall				
Note				
Configuration of filter rules on IPv4 level. Tags: atp.Status=candidate				
Base				
ARObject, NetworkLayerRule				
Aggregated by				
FirewallRule.networkLayerRule				
Attribute	Type	Mult.	Kind	Note
checksumVerification	Boolean	0..1	attr	Defines whether a Ipv4 header checksum verification is performed or not. Tags: atp.Status=candidate
destinationIpAddress	Ip4AddressString	0..1	attr	Filter to match packets with the destination IPv4 address. Tags: atp.Status=candidate
destinationNetworkMask	Ip4AddressString	0..1	attr	Filter to match packets with the destination IPv4 address range. The destinationIpAddress with the destinationNetworkMask defines the IP address range. Tags: atp.Status=candidate
differentiatedServiceCodePoint	PositiveInteger	0..1	attr	Filter to match packets with a DSCP value. Tags: atp.Status=candidate
doNotFragment	Boolean	0..1	attr	Filter to match packets that have the doNotFragment bit in the Header set. Tags: atp.Status=candidate





Class	Ipv4Rule			
explicitCongestionNotification	PositiveInteger	0..1	attr	Filter to match packets with a ECN code point. Tags: atp.Status=candidate
icmpRule	IcmpRule	0..1	aggr	Configuration of filter rules for ICMP (Internet Control Message Protocol). Tags: atp.Status=candidate
internetHeaderLength	PositiveInteger	0..1	attr	Filter to match packets with a minimum ipv4 header length. Tags: atp.Status=candidate
moreFragments	Boolean	0..1	attr	Filter to match packets that have the moreFragments flag in the Header set. Tags: atp.Status=candidate
protocol	PositiveInteger	0..1	attr	Filter to match packets with a IP protocol number . Tags: atp.Status=candidate
sourceIpAddress	Ip4AddressString	0..1	attr	Filter to match packets with the source IPv4 address. Tags: atp.Status=candidate
sourceNetworkMask	Ip4AddressString	0..1	attr	Filter to match packets with the source IPv4 address range. The sourceIpAddress with the sourceNetworkMask defines the IP address range. Tags: atp.Status=candidate
ttlMax	PositiveInteger	0..1	attr	Filter to match packets with a maximum ttl value (TimeToLive defines the lifetime of data on the network). Tags: atp.Status=candidate
ttlMin	PositiveInteger	0..1	attr	Filter to match packets with a minimum ttl value (TimeToLive defines the lifetime of data on the network). Tags: atp.Status=candidate

Table A.8: Ipv4Rule

Class	Ipv6Rule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules on IPv6 level. Tags: atp.Status=candidate			
Base	ARObject , NetworkLayerRule			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	Type	Mult.	Kind	Note
destinationIpAddress	Ip6AddressString	0..1	attr	Filter to match packets with the destination IPv6 address. Tags: atp.Status=candidate
destinationNetworkMask	Ip6AddressString	0..1	attr	Filter to match packets with the destination IPv6 address range. The destinationIpAddress with the destinationNetworkMask defines the MAC address range. Tags: atp.Status=candidate
flowLabel	PositiveInteger	0..1	attr	Filter to match packets with a defined flow label. Tags: atp.Status=candidate
hopLimit	PositiveInteger	0..1	attr	Filter to match packets with a minimum hop limit. Tags: atp.Status=candidate





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

Table A.9: Ipv6Rule

Class	ModeDeclaration			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
Base	<i>ARObject</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , Referrable			
Aggregated by	<i>AtpClassifier.atpFeature</i> , 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.10: ModeDeclaration

Class	ModeDeclarationGroup			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. Tags: atp.recommendedPackage=ModeDeclarationGroups			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>CollectableElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , Referrable , <i>UploadableDesignElement</i> , <i>UploadablePackageElement</i>			
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.





Class	ModeDeclarationGroup			
mode Declaration	ModeDeclaration	*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=modeDeclaration.shortName, mode Declaration.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime

Table A.11: ModeDeclarationGroup

Class	<i>NetworkLayerRule</i> (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules on the Network layer Tags: atp.Status=candidate			
Base	<i>ARObject</i>			
Subclasses	Ipv4Rule , Ipv6Rule			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	Type	Mult.	Kind	Note
-	-	-	-	-

Table A.12: NetworkLayerRule

Class	PayloadBytePatternRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of a generic firewall rule that defines the individual bytes of a message that shall match. Tags: atp.Status=candidate			
Base	<i>ARObject</i>			
Aggregated by	FirewallRule.payloadBytePatternRule			
Attribute	Type	Mult.	Kind	Note
payloadByte PatternRulePart	PayloadBytePattern RulePart	*	aggr	Configuration of bytes in the message, Tags: atp.Status=candidate

Table A.13: PayloadBytePatternRule

Class	<i>Referrable</i> (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	<i>ARObject</i>			
Subclasses	<i>AtpDefinition</i> , <i>BswDistinguishedPartition</i> , <i>BswModuleCallPoint</i> , <i>BswModuleClientServerEntry</i> , <i>BswVariableAccess</i> , <i>CouplingPortTrafficClassAssignment</i> , <i>CplusplusImplementationDataTypeContextTarget</i> , <i>DiagnosticEnvModeElement</i> , <i>EthernetPriorityRegeneration</i> , <i>ExclusiveAreaNestingOrder</i> , <i>HwDescriptionEntity</i> , <i>ImplementationProps</i> , <i>ModeTransition</i> , <i>MultilanguageReferrable</i> , <i>NmNetworkHandle</i> , <i>PncMappingIdent</i> , <i>SingleLanguageReferrable</i> , <i>SoConIPdulIdentifier</i> , <i>SocketConnectionBundle</i> , <i>SomeipRequiredEventGroup</i> , <i>TimeSyncServerConfiguration</i> , <i>TpConnectionIdent</i>			
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: xml.sequenceOffset=-90

Table A.14: Referrable

Class		SomeipProtocolRule		
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of SOME/IP firewall rules Tags: atp.Status=candidate			
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. Tags: atp.Status=candidate
length Verification	Boolean	0..1	attr	Defines whether length verification is performed or not. Tags: atp.Status=candidate
majorVersion	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the majorVersion in the SOME/IP header matches. Tags: atp.Status=candidate
messageType	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the messageType in the SOME/IP header matches. Tags: atp.Status=candidate
methodId	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the methodId in the SOME/IP header matches. Tags: atp.Status=candidate
protocolVersion	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the protocolVersion in the SOME/IP header matches. Tags: atp.Status=candidate
returnCode	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the returnCode in the SOME/IP header matches. Tags: atp.Status=candidate
serviceInterface Id	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the serviceInterfaceId in the SOME/IP header matches. Tags: atp.Status=candidate

Table A.15: SomeipProtocolRule

Class	SomeipSdRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of SOME/IP Service Discovery firewall rules Tags: atp.Status=candidate			
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. Tags: atp.Status=candidate
eventGroupId	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the eventGroupId in the SOME/IP header matches. Tags: atp.Status=candidate
maxMajorVersion	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. Tags: atp.Status=candidate
maxMinorVersion	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. Tags: atp.Status=candidate
minMajorVersion	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. Tags: atp.Status=candidate
minMinorVersion	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. Tags: atp.Status=candidate
serviceInstanceId	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the serviceInstanceId in the SOME/IP header matches. Tags: atp.Status=candidate
serviceInterfaceId	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the serviceInterfaceId in the SOME/IP header matches. Tags: atp.Status=candidate

Table A.16: SomeipSdRule

Class	StateDependentFirewall			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Firewall rules that are defined in a firewall state Tags: atp.Status=candidate atp.recommendedPackage=StateDependentFirewallRules			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDesignElement, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note





Class	StateDependentFirewall			
defaultAction	FirewallActionEnum	0..1	attr	This attribute defines a defaultAction in case that the VehicleMode is not yet set. Tags: atp.Status=candidate
firewallRule Props (ordered)	FirewallRuleProps	*	aggr	Collection of firewall rules that apply in the vehicle mode Tags: atp.Status=candidate
firewallState	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.Status=candidate InstanceRef implemented by: FirewallStateInFirwall StateSwitchInterfaceInstanceRef

Table A.17: StateDependentFirewall

Class	TcpRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of TCP filter rules. Tags: atp.Status=candidate			
Base	<i>ARObject</i> , TransportLayerRule			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	Type	Mult.	Kind	Note
numberOfParallelTcpSessions	PositiveInteger	0..1	attr	This attribute defines the maximal number of TCP Sessions that are allowed to be established. Tags: atp.Status=candidate
stateManagementBasedOnTcpFlags	Boolean	0..1	attr	This attribute defines whether the StateManagement is based on TCP flags or not. Tags: atp.Status=candidate
timeoutCheck	PositiveInteger	0..1	attr	This attribute defines the TCP Session timeout in seconds Tags: atp.Status=candidate

Table A.18: TcpRule

Class	<i>TransportLayerRule</i> (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules on Transport Layer level. Tags: atp.Status=candidate			
Base	<i>ARObject</i>			
Subclasses	TcpRule , UdpRule			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	Type	Mult.	Kind	Note
checksumVerification	Boolean	0..1	attr	Defines whether checksum verification is performed or not. Tags: atp.Status=candidate





Class	TransportLayerRule (abstract)			
maxDestinationPortNumber	PositiveInteger	0..1	attr	Filter to match packets with the maximum destination UDP/TCP port number. Tags: atp.Status=candidate
maxSourcePortNumber	PositiveInteger	0..1	attr	Filter to match packets with the maximum source UDP/TCP port number. Tags: atp.Status=candidate
minDestinationPortNumber	PositiveInteger	0..1	attr	Filter to match packets with the minimum destination UDP/TCP port number. Tags: atp.Status=candidate
minSourcePortNumber	PositiveInteger	0..1	attr	Filter to match packets with the minimum source UDP/TCP port number. Tags: atp.Status=candidate

Table A.19: TransportLayerRule

B Platform Extension API (normative)

The focus of the APIs in this section are for OEM-specific platform extensions. The abstraction of the interfaces is lower which could lead to a higher machine dependency.

C Interfaces to other Functional Clusters (informative)

C.1 Overview

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 informative guidelines how the interaction between Functional Clusters looks like, by clustering the relevant requirements of this document to describe Inter-Functional Cluster (IFC) interfaces. In addition, the standardized public interfaces which are accessible by user space applications (see chapters 8 and 9) can also be used for interaction between Functional Clusters.

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 the interfaces are up to the platform provider. Additional interfaces, parameters and return values can be added.

C.2 Interface Tables

No content.

D History of Constraints and Specification Items

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

D.1 Traceable item history of this document according to AUTOSAR Release R22-11

D.1.1 Added Specification Items in R22-11

Number	Heading
[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]	





Number	Heading
[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]	





Number	Heading
[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]	





Number	Heading
[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]	Generated header files for FirewallStateSwitchInterface
[AP_SWS_Fw_-81001]	Enumeration for FirewallStateSwitchInterface
[AP_SWS_Fw_-81002]	Definition of enumerators of FirewallStateSwitchInterface
[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]	



△

Number	Heading
[AP_SWS_Fw_-82008]	

Table D.1: Added Specification Items in R22-11

D.1.2 Changed Specification Items in R22-11

none

D.1.3 Deleted Specification Items in R22-11

none

D.2 Constraint and Specification Item History of this document according to AUTOSAR Release 23-11

D.2.1 Added Specification Items in R23-11

[\[AP_SWS_Fw_31001\]](#) [\[AP_SWS_Fw_31002\]](#) [\[AP_SWS_Fw_61000\]](#)

D.2.2 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]

D.2.3 Deleted Specification Items in R23-11

none