Document Title	Explanation of QoS Policies in the scope of SOME/IP
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1143

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R25-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2025-11-27	R25-11	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

1 Introduction

Quality of Service (QoS) is a set of network technologies and mechanisms to manage and prioritize network traffic. Although the original focus was on communication itself, middleware protocols try to achieve certain QoS functionalities and add additional properties, such as data persistence, data filtering and resource management.

1.1 Objectives

This document provides an overview of QoS policies commonly used by communication middlewares, such as DDS and Zenoh, and explains their relevance to automotive communications. It establishes a shared knowledge base and demonstrates which of the relevant QoS policies can be implemented using existing AUTOSAR features.

1.2 Scope

This document shows which policies can be implemented with existing AUTOSAR features according to [1, SOME/IP Protocol Specification], [2, SOME/IP Service Discovery Protocol Specification] and related documents.

2 Definition of terms and acronyms

2.1 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
DDS	Data Distribution Service
E2E	End-to-End
QoS	Quality of Service
SOME/IP	Scalable service-Oriented MiddlewarE over IP
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
Zenoh	Zero Overhead Network Protocol

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

2.2 Definition of terms

Terms:	Description:
Jitter	Jitter is the variation of latency.
Latency	Latency is the time it takes for data to pass from one point to another. For example, it could be the time between network endpoints or between APIs.

Table 2.2: Definition of terms in the scope of this Document

3 Related Documentation

3.1 Input documents & related standards and norms

[1] SOME/IP Protocol Specification
AUTOSAR_FO_PRS_SOMEIPProtocol

[2] SOME/IP Service Discovery Protocol Specification
AUTOSAR_FO_PRS_SOMEIPServiceDiscoveryProtocol

[3] E2E Protocol Specification
AUTOSAR_FO_PRS_E2EProtocol

[4] Requirements on Health Monitoring
AUTOSAR_FO_RS_HealthMonitoring

[5] Specification of Automotive API Gateway
AUTOSAR_AP_SWS_AutomotiveAPIGateway

[6] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

4 Different Types of QoS

This section provides an overview of QoS policies, which are available in communication protocols like DDS or Zenoh.

These policies are divided into five groups: Data Transmission, Data Persistence, Data Management, Resource Management, and Miscellaneous.

Many more policies are available for different types of protocols. However, this document will focus on the most relevant ones.

4.1 Data Transmission

The primary focus of QoS is data transmission. It covers topics such as traffic prioritization and delivery guarantees, as well as latency, latency jitter and bandwidth. While other types of QoS are primarily handled at upper software layers, QoS policies for data transmission are also handled at network and data link layer.

4.1.1 Reliability Policy

Specifies mechanisms and strategies to ensure data transmission and consistency. This includes error detection, correction or retransmission techniques.
This policy is designed to ensure that data is delivered accurately and without loss. One example of this is delivery guarantee types, such as 'at most once', 'at least once' and 'exactly once'.

4.1.2 Interference/Priority Policy

Specifies mechanisms to avoid interference and to prioritize data transmissions. This allows critical data to be prioritized over less important data, ensuring that data with higher priority is transmitted in time.
This policy is crucial for applications requiring timely and predictable data delivery and can be used to realize latency requirements.

On network level, this policy is related to the IEEE802.1Q standard, which specifies the management of network traffic.

4.2 Data Persistence

Data persistence includes mechanisms related to the availability and lifetime of data. These QoS policies are mainly handled at the presentation or application layer.

4.2.1 Durability Policy

Specifies how long data should be retained on provider side, ensuring data persistence across system restarts and failures. Also specifies if already existing data shall be available for new subscribers/consumers.

4.2.2 History Policy

Specifies the maximum number of data samples to be buffered on either provider or consumer side.

4.2.3 Lifespan Policy

Specifies the retention and validity of data over a specified period. It defines the duration for which data remains valid and can be used by subscribing nodes.
A time-sychronization between provider and consumer is required to realize this policy.

4.3 Data Management

Data management includes policies about the data itself. This includes ownership as well as the handling of data types and the relationship between different data samples. These QoS policies are all handled at the presentation or application layer.

4.3.1 Ownership Policy

Specifies whether data can be owned by one or multiple providers.

4.3.2 Presentation Policy

Specifies how data is presented to the consumers. This allows data to be presented to the consumer dependent on other data. For example, data from different sources can be provided at once to the consumer.

4.3.3 Type Consistency Policy

Specifies rules for determining whether and how the data types used by the provider and consumer must match.

4.4 Resource Management

Resource management includes policies for the use of resources on both the provider and consumer sides, e.g. buffer sizes.

4.4.1 Resource Limits Policy

Manages the allocation and usage of system resources. This policy is used to ensure that resources such as memory, CPU, etc. are used efficiently and within predefined limits.

4.5 Miscellaneous

4.5.1 Liveliness Policy

Specifies mechanisms and strategies used to monitor and confirm the presence of network nodes and/or applications.
This policy can be used to verify that the required entities are alive and functional.

4.5.2 User Data Policy

Specifies the exchange of user-defined data. The middleware provides the functionality for exchanging this data. It is the responsibility of the applications to set and process the data.

4.5.3 Filtering Policy

Specifies how data is filtered on the provider and/or consumer side at runtime. This can be used to prevent unwanted data from being sent or processed. The filtering can either be based on the data content itself or on time.

This policy has two use cases. The first is to reduce bandwidth and resource usage during runtime, which can be done on the provider or consumer side.
The second use case is security-related. Any invalid or unauthorized data received shall be dropped and not processed.

5 Realizing QoS Policies with SOME/IP

This section provides an overview of how the aforementioned policies can be implemented using SOME/IP. It also states which policies are relevant and supported.
A comprehensive status overview is available in chapter 6.

5.1 Reliability

Reliability encompasses multiple mechanisms. These include error detection, error correction, and data retransmission. The necessity of these mechanisms is determined by the data and its intended use case.

TCP/UDP

Both TCP and UDP offer error detection mechanisms at the network level.
TCP uses a checksum to detect transmission errors, and acknowledgements (ACKs) along with sequence numbers to detect packet loss and ensure packets are received in the correct order. If a packet is lost (indicated by a missing ACK), TCP requests a re-transmission of the packet, ensuring reliable communication. However, it should be noted that error recovery is typically incompatible with low-latency requirements.
UDP uses checksums to identify transmission errors. If the calculated checksum does not match the expected checksum, an error is indicated. However, UDP does not have built-in mechanisms for error recovery.

E2E

The E2E protection mechanisms used by SOME/IP ensure the integrity of message transmission. A variety of profiles are available to support different types of use cases.

Following faults can be detected with E2E:

Fault	Main safety mechanism
Repetition of information	Counter
Loss of information	Counter
Delay of information	Counter
Insertion of information	Data ID
Masquerading	Data ID, CRC
Incorrect addressing	Data ID
Incorrect sequence of information	Counter
Corruption of information	CRC
Asymmetric information sent from a sender to multiple receivers	CRC
Information from a sender received by only a subset of the receivers	Counter
Blocking access to a communication channel	Counter

Table 5.1: Detectable Faults with E2E
List of faults that can be detected with E2E.

For E2E, only consumers will be aware of errors. The provider will not be automatically informed.

See the [3, E2E Protocol Specification] for further information.

Methods

Methods (excluding fire-and-forget) can be used for error detection, as the response can be used to return structured error information to the calling application. SOME/IP utilizes standardized error codes and optional error descriptions within the method response payload to indicate the type and details of any errors encountered during processing. It is the responsibility of the calling application to process the error response, interpret the error code and description, and take appropriate action.

See [1, SOME/IP Protocol Specification], chapter "4.2.6 Error Handling", for further information.

System Health Monitoring

System Health Monitoring (SHM) provides a health monitoring on system level. It focuses on system wide coordination of error handling across multiple platforms on multiple controllers and machines.

See [4, Requirements on Health Monitoring] for further information.

Relevance and Status

Reliability is important for various automotive communication use cases. There are many existing features that can be used to realize these use cases. It is important to note that the mechanisms are designed primarily for unicast traffic, but this is sufficient.

5.2 Interference / Priority

TSN specifies mechanisms in IEEE802.1Q to avoid interference and to comply with latency requirements. The SOME/IP configuration provides timing-relevant parameters and requirements (e.g., bandwidth, burst size, frame size) that can be used to configure the network setup (switches, etc.) accordingly. If hard latency requirements must be met, this must be done in the physical and data link layer and can not be realized in higher layers.

Relevance and Status

It is crucial to prevent interference and ensure latency and priority for automotive communication. Multiple mechanisms are available to support these requirements, primarily those specified by IEEE 802.1Q.

5.3 Durability

The realization of this policy involves both the provider and consumer, depending on the communication pattern (Events, Methods, Fields). Data buffering is typically implemented on the provider side to support durability.

Events

Consumers will only get event data that is sent after the subscription. It is not possible to receive data that was set before the subscription.

If data that was set before the consumer subscribes is to be accessible, fields should be used.

Fields

With Fields, consumers can access data that was set before they subscribed to it. They will also receive this value as an initial event upon subscribing. Therefore, there is no need to wait for the data to be set again in order to receive it. See [1, SOME/IP Protocol Specification], chapter "4.2.5 Fields", for further information.

Methods

While not directly implemented through buffering, durability considerations apply to methods. The reliability of method responses depends on mechanisms like request correlation and timeouts to handle potential provider failures before the response is sent.

Relevance and Status

Although this policy is not that relevant to communication quality itself, it can be implemented through the use of fields.

5.4 History

The realization of this policy depends on the communication pattern (Events, Methods, Fields). This kind of data buffering can be related to both provider and/or consumer side.

Events

The idea behind events is to always transmit the latest value. Therefore, having a history for events might not be a use case. If a history is really required, this could be handled by the consumer application. But a mechanism would be required that allows the consumer to check the history status (e.g., size).

The only situation in which having a history on the provider side could be useful would be if the consumer restarts. However, since the consumer will need to be completely reinitialized anyway, providing a history on the provider side might not be helpful. At least, the cost of performance due to the additional buffering is not worth it.

Fields

For fields, a history could be useful. At the moment, this would need to be realized on application level on both sides.

Methods

For methods, having a history is not relevant, as each response is directly related to a request.

Relevance and Status

This policy is not really relevant to the quality of communication itself. If necessary, history handling should be implemented at the application level.

5.5 Lifespan

The realization of this policy depends on the communication pattern (Events, Methods, Fields).

Events

For events, the update interval is the lifespan. If event data should become outdated before the next event is received, an additional handling has to be implemented by the consumer application.

Fields

For fields, the lifespan of the data needs to be handled by the provider application. If a lifespan monitoring is required, a timestamp needs to be added to the payload.

Methods

Methods support a timeout mechanism, which can be used for kind of a lifespan handling.

Relevance and Status

This policy is not relevant to the quality of communication itself. If necessary, lifespan handling should be implemented at the application level.

5.6 Ownership

Because different providers of the same SOME/IP service must use different Instance IDs, multi-ownership is not possible using SOME/IP.

For fields, it is possible that multiple applications are setting the field value. But in this case, the ownership will still remain with the provider.

Relevance and Status

In SOME/IP, ownership is handled by design.

5.7 Presentation

Data is always presented as it is received. If data from different sources is to be presented simultaneously, or if the data is to be changed in any way, this should be done at the application level.

AUTOSAR already provides some components for various use cases of that kind. For more information, see A.1.

Relevance and Status

This policy is not directly related to the quality of communication. If necessary, these mechanisms will be implemented at the application level.

5.8 Type Consistency

Since the data types are defined in the system model for the automotive commmunication, no extra rules should be required for runtime checking and thus this policy would not be relevant. It is also possible to extend the payload format while maintaining compatibility. See [1, SOME/IP Protocol Specification], chapter "4.3 Compatibility Rules for Interface Version", for further information.

If the data types require breaking changes, the interface version of a service should be updated.
This allows for backward compatibility. For example, the provider could offer the service with both the old and new data types. Consumers can then subscribe to the appropriate version.

The only reason dynamic type handling at runtime would be useful is for code-first or rapid prototyping use cases. In those cases, the service might not be fully defined yet and could change regularly.

Relevance and Status

Ensuring type consistency is key to ensuring proper communication. Currently, this is addressed by a comprehensive system model and interface service versions. However, further mechanisms could be useful in the context of a code-first approach or for rapid prototyping.

5.9 Resource Limits

It is possible to reduce the resource usage by configuring queue sizes, etc. Furthermore, the system design defines the resource usage, especially for periodic communication, which could be used to calculate the resource consumption for involved ECUs.

Relevance and Status

This type of policy implicitly impacts communication quality. However, there are many resource-related configuration parameters available to optimize the system.

5.10 User Data

At service discovery, the Configuration Option can be used to exchange additional data. See the [2, SOME/IP Service Discovery Protocol Specification], chapter "5.1.2.4.1 Configuration Option", for further information.

But there is no possibility to attach additional user data to the regular data stream (e.g., Events).

Relevance and Status

The quality of communication is not directly affected by the presence of user data, it is just a comfort feature. Therefore it is not relevant for the communication quality.

5.11 Liveliness

The liveliness of data can be checked by having a cyclic event transmission with deadline monitoring (E2E). Depending on the setup, this could also be used to verify the liveliness of the related application. If there are no events available or only non-periodic events, an additional heartbeat event could be added to create a liveliness mechanism.

The liveliness of services can be checked by using the Service Discovery. If configured, cyclic offers with lifetime (TTL) can be used to monitor whether providers are still alive and functional. On the other hand, service subscriptions can be used to check that at least one consumer is operational.

Relevance and Status

The liveliness policy is useful for determining the overall status of the system. Periodic communication can be used for this purpose. If only non-periodic communication is available, an additional liveliness mechanism must be implemented at the application level.

5.12 Filtering

There is no filter mechanism available to reduce the bandwidth and resource usage at runtime. If the data is to be provided in smaller parts or at different intervals, this must be defined in the system model and cannot be changed at runtime. However, the system must be designed to handle the maximum possible bandwidth. Therefore, the benefit of filtering is questionable.

For security, filtering on address information is possible on different protocol layers. For example, the IEEE 802.1Q standard defines multiple filters for raw Ethernet.

Relevance and Status

Although not provided by SOME/IP itself, multiple mechanisms are available for runtime filtering in the scope of security.

There are no mechanisms available for filtering to reduce bandwidth and resource usage dynamically at runtime. This is because effective automotive communication relies on good service design and appropriate timings, eliminating the need for runtime filtering.

6 Status

Policy	Status	Description
Reliability 5.1	supported	
Interference / Priority 5.2	supported	
Durability 5.3	supported	
History 5.4	not relevant	if required, it should be handled on application level
Lifespan 5.5	not relevant	if required, it should be handled on application level
Ownership 5.6	supported	
Presentation 5.7	not relevant	if required, it should be handled on application level
Type Consistency 5.8	partially supported	type consistency is ensured by a comprehensive system model; but no support for runtime checks
Resource Limits 5.9	supported	
User Data 5.10	not relevant	if required, it should be handled on application level
Liveliness 5.11	partially supported	supported for periodic communication, for non-periodic communication it needs to be handled on application level
Filtering 5.12	partially supported	only supported for security, not for bandwidth and resource optimization

Table 6.1: Status of QoS in SOME/IP

Current status of common QoS policies in SOME/IP.

A Appendix

A.1 Realization of Presentation Policy in Adaptive

Features related to the Presentation Policy (5.7) shall be handled at application layer. The Adaptive Platform already provides components for such use cases. For example the Automotive API Gateway [5] or the Pass-Through-Composition via PassThrough-SwConnectors [6].