ETSI GS MEC 045 V3.1.1 (2024-03)

GROUP SPECIFICATION

Multi-access Edge Computing (MEC); QoS Measurement API

Disclaimer

Reference

DGS/MEC-0045v311QoSMeasure

Keywords

API, measurement, MEC, QoS, service

Contents

Intellectual Property Rights

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Multi-access Edge Computing (MEC).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document focuses on introducing QoS Measurement service to provide network performance related metrics including predictive QoS provided by AI/ML components if available. It describes the reference scenarios and sequence diagrams and defines the data model and API. The present document carefully considers the relevant work of other SDOs (e.g. 3GPP, 5GAA, etc.) and all relevant work done in ETSI.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI GS MEC-IEG 006: "Mobile Edge Computing; Market Acceleration; MEC Metrics Best Practice and Guidelines".

[2] ETSI GS MEC 009: "Multi-access Edge Computing (MEC); General principles, patterns and common aspects of MEC Service APIs".

[3] OMA-TS-REST-NetAPI-ZonalPresence-V1-0-20160308-C: "RESTful Network API for Zonal Presence".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] 3GPP TR 22.847: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on supporting tactile and multi-modality communication services; Stage 1 (Release 18)".

[i.2] 3GPP TS 29.522: "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; 5G System; Network Exposure Function Northbound APIs; Stage 3 (Release 18)".

[i.3] ETSI GR MEC 018: "Mobile Edge Computing (MEC); End to End Mobility Aspects".

[i.4] 3GPP TS 23.501: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 18)".

[i.5] ETSI GS MEC 013: "Multi-access Edge Computing (MEC); Location API".

[i.6] ETSI GS MEC 030: "Multi-access Edge Computing (MEC); V2X Information Services API".

[i.7]	ETSI GR MEC 001: "Multi-access Edge Computing (MEC); Terminology".
[i.8]	OpenAPI™ Specification.
[i.9]	ETSI GS MEC 010-2: "Multi-access Edge Computing (MEC); MEC Management; Part 2: Application lifecycle, rules and requirements management".

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GR MEC 001 [i.7] and the following apply:

QMS	QoS Measurement Service

4 Overview

The present document specifies QMS API that enables the collection and provision of the network performance related metrics for helping the API consumers monitor, predict and/or improve the network performance.

Clause 5 presents the reference scenarios for QoS measurement and also describes the relevant information flows.

Clause 6 describes the data models that can be exchanged over the QoS Measurement APIs, which provide detailed descriptions of all information elements used for QoS Measurement.

Clause 7 defines the actual QoS Measurement APIs providing detailed information of how information elements are mapped into a RESTful API design.

5 Description of the services (informative)

5.1 Reference scenarios

5.1.1 QoS metrics

QoS is used to measure and control network service quality. Following the rapid development of the mobile network, the services are becoming much more diverse, and the network traffic is flooding. Consequently, it becomes more and more challenging to handle severe network congestion, increasing packet delay, and even packet loss that lead to the degradation of the quality of experience. From the operation and maintenance cost perspective, it is not feasible to solve or mitigate the problems only by increasing bandwidth. A set of QoS metrics are defined to describe the differentiated requirements of the network quality for different network services, and specific QoS policies would be implemented according to the service characteristics to improve the overall usage of network resources.

The important QoS metrics include but not limited to:

- Network delay: It is defined as the latency for a data packet to travel across the network from one communication endpoint to another. Some services are sensitive to network delays. For example, Virtual Reality (VR) games can provide a better experience comparing to traditional games, and most immersive VR games require the end-to-end delay to be less than 20 ms. The VR game participants may feel inconveniences if the network delay increases.

- Jitter: Different packets of a service flow delivered in the network have different delay values. The deviation of delay is called jitter. As one of the most crucial factors to quality of service, jitter is mainly caused by queueing time. jitter cannot be avoided and has no impact if the value is small enough. Many services, like eXtended Reality (XR), are sensitive to jitter. Caching packets may be leveraged to overcome the exceeding jitter.

- Packet loss: If a small number of packets are lost, services are not severely affected. But severe packet loss could impact transport efficiency. Hence, it is worth focusing on the statistical metric of packet loss - packet loss ratio, which refers to the percentage of lost packets to the overall transmitted packets during network transmission.

- Throughput: It refers to the bit numbers delivered from one communication endpoint to another in a fixed period of time (e.g. 1 second). It can be defined in downlink or uplink. The unit of throughput is e.g. bit/s or Kbit/s. The throughput's influence on applications is described in ETSI GS MEC-IEG 006 [1].

5.1.2 Synchronization delay

In addition to measuring the QoS metrics for a single flow, the emergence of multi-modal services brings the requirements of multi-flow QoS guarantee for networks. Multi-modal service refers to the combination of multiple data flows to provide users with an immersive experience. In a VR game, a user may wear a VR helmet (visual, auditory) and VR gloves (tactile) to receive game information. Meanwhile, the game server (MEC application) needs to collect environment information about the user, for instance, collecting sound by using a microphone, collecting user action information by using a camera, and collecting intensity of user actions by using a sensor.

For multi-modal service, the QoS metrics of a single data flow are not enough to fully reflect the overall quality of service. It is necessary to provide a manner to simultaneously measure and monitor the synchronization delay, i.e. the network delay difference between two flows. Sometimes, all the audio, video, and tactile flows work very well, but the synchronization delays between them may still make users feel inconveniences. For example, an explosion in a VR game (involving an audio stream, video stream, and tactile stream) can only make the game experience more realistic when synchronization delays of multiple streams with dependencies are in a reasonable range. In this case, a synchronization threshold may be provided by the application. If the synchronization threshold is exceeded, a data asynchronization event will be notified to the application. The synchronization threshold is described in 3GPP TR 22.847 [i.1].

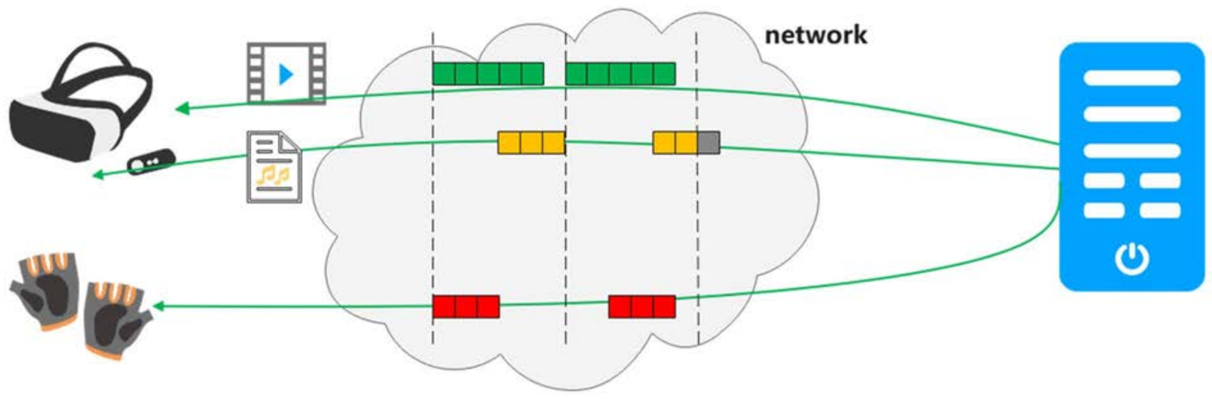

Figure 5.1.2-1: Downlink example for Multi-modal service

5.1.3 QMS introduction

A QMS is a MEC service used to measure the quality of an edge service by monitoring the data flows associated with one or more terminal devices accessing the edge service. A consumer of the QMS may be a MEC application or the MEP. Consumers can invoke QMS APIs through the Mp1 reference point to query QoS information, create QoS monitoring tasks, or subscribe to QoS monitoring results. Considering the system burden, part of the data flows associated with a service, rather than all data flows, could be measured to reflect the network performance.

The QMS could leverage the way provided by the MEC access network to get the network performance related to a MEC application. The technologies of MEC access network are various e.g. 5G, WLAN, Fibre, Cable. Not only the 5G network, the QMS can work with any MEC access network, as long as they have a method to measure the performance. The customers of the QMS do not need to understand the detailed network topology or the type of MEC access network. For example, if 5G UEs are accessing a video service provided by a MEC instance that wants to measure 5G network performance, the MEC instance can invoke the QMS API to pass the active UE list to QMS and create a measuring task. After that, the QMS could utilize the "QoS-Monitoring" capability (as described in 3GPP TS 29.522 [i.2]) exposed by NEF to get the 5G network performance.

NOTE: MEC access network is a terminology in the MEC context, which represents the network working between a terminal device and the edge network at which a MEC host is located. For 5G, the MEC access network consists of a radio network, a core network and a transport network connecting them.

5.1.4 Use cases

5.1.4.1 Assistance of application deployment and migration

The edge service running at different MEC hosts may have different performances due to the difference in deployment locations and access networks. For example, some MEC applications may be deployed at the city level to provide services for multiple enterprises at the same time. Other MEC applications may be deployed on-premises because of security or low latency requirements. The various access networks and the heterogeneous infrastructure of edge computing also bring differentiated network service capabilities.

It is implementation dependent on how to measure or predict the network quality a MEC host can provide. A possible way is that each MEC host could instantiate an application instance communicating with one or more test UEs. And each MEP invokes the QMS APIs to create a task for QoS measurement and obtains a result regarding network delay under the Best-Effort model by monitoring the QoS information of test flows between the test UEs and the test MEC application instance.

Before an application is deployed at a MEC host, the network QoS of different MEC hosts needs to be evaluated to determine whether the MEC host can meet its QoS requirements. According to the requirements of ETSI GS MEC 010-2 [i.9], when deploying an application, the application provider can use the optional "appLatency" attribute in the AppD to describe the maximum delay that the MEC application can tolerate. QMS can be used to evaluate the network latency of the target MEC host. Based on that, the MEO can easily select an appropriate MEC host to deploy the application.

UE's movement may lead to service discontinuity (two scenarios regarding mobility in the MEC system are described in clause 4.1.2 of ETSI GR MEC 018 [i.3]). If a UE moves out of the coverage area of the source MEC host, the MEC system may need to migrate the MEC application serving the UE from the source MEC host to the target MEC host. Or the MEC system may transfer the UE context to the target application instance if the application has been instantiated at the target MEC host already. If more than one MEC hosts are candidate, QMS can measure network delay values to assist in selecting the most appropriate target MEC host for delay-sensitive applications.

5.1.4.2 Monitoring network performance

MEC application instances (e.g. server applications) can use QMS to monitor QoS in various forms. When noticing the service quality for some of its subscribers (e.g. client applications residing in the terminals) deteriorates, the MEC application as the consumer can provide the subscriber IDs (e.g. MSISDN, other anonymous ID) to enable the QMS to monitor the network performance for these target subscribers. If the MEC application wants to measure the QoS of some specific flows, it could provide a set of flow information (e.g. 5-tuple) to QMS.

A MEC application can also enable a QoS monitoring task by providing a flow sampling rate rather than concrete subscriber information. This method avoids security risks caused by the transmission of private information. Different sampling rates can help the system flexibly adjust the load of monitoring tasks as required. For example, the consumer could ask the QMS to monitor "x" percent of flows associated with the target application by providing a sampling rate as "x". The QMS could identify the target flows by checking 5-tuple or 7-tuple, e.g. the IP address of the target edge server (target application instance). Then, The QMS selects "x" percent of target flows to monitor and measure the QoS performance.

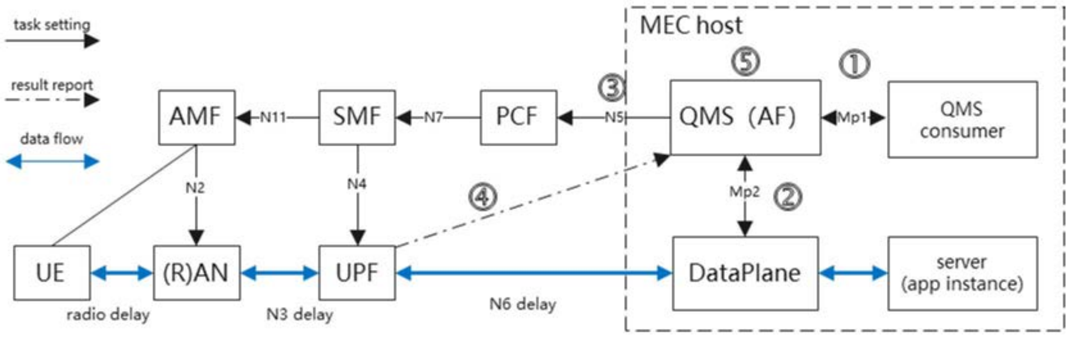

Figure 5.1.4.2-1: Monitoring data flows according to sampling rate in 5G network

The service consumer provides a sampling rate for network performance monitoring, as illustrated in Figure 5.1.4.2-1, which consists of the following steps:

1) A service consumer (an application instance) sends a request carrying a sampling rate and server IP address to create a monitoring task. Compared to providing subscriber IDs, this method applies to scenarios in which the consumer does not know the specific UE information.

2) The QMS enables the data plane to filter the data flows by the server IP address which could be source or destination IP address. After that, the QMS can get a list of UEs who are accessing the target server. It needs to select part of the UEs as monitoring targets according to the sampling rate received in step 1).

3) The QMS producer triggers 5GS to enable the QoS monitoring process (see 3GPP TS 23.501 [i.4]).

4) After measurement, the UPF reports the results to the QMS producer.

5) The QMS could converge the results and notify the consumer of the final result.

NOTE 1: The N6 delay that is not included in QoS monitoring result normally is very small and could be neglected when the MEP and UPF are co-deployed.

NOTE 2: If the MEC host locates in the untrusted domain, the QMS as an AF should communicate with 5GS through NEF.

5.1.4.3 Measuring QoS by area

The QMS can measure the QoS according to the area information provided by the consumer. In that way, the measuring task will be enabled if the measuring targets stay in a specific area. The consumer could leverage the QMS to monitor the network performance of some specific areas and get a notification if network degradation happens.

In the autonomous driving scenario, a V2X application deployed at the edge collects environmental information (e.g. vehicle flow rate, road status) from the cars or the RSUs to make a decision for route planning. Generally, this service has high requirements on QoS for handling the newest information promptly. The candidate routes may span multiple areas whose network performances are different. The QMS needs to provide a service enabling the QoS measuring task when the terminal devices stay in specific areas, but for this use case, the QMS also needs to merge all the measuring results related to one area to get the network performance of this area.

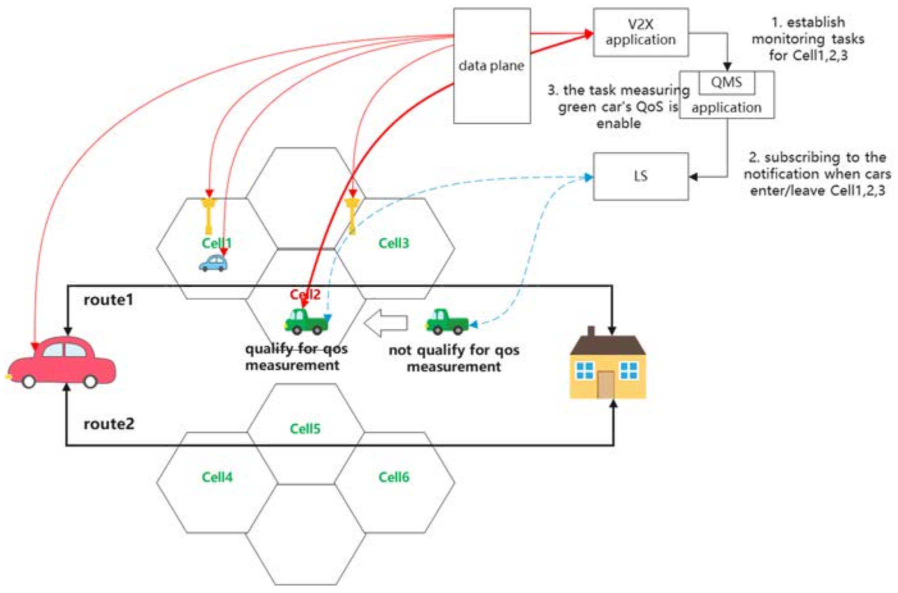
data plane
V2X application
1. establish monitoring tasks for Cell1,2,3
QMS application
3. the task measuring green car's QoS is enable
LS
2. subscribing to the notification when cars enter/leave Cell1,2,3
Cell1
Cell2
Cell3
Cell4
Cell5
Cell6
route1
route2
qualify for qos measurement
not qualify for qos measurement

Figure 5.1.4.3-1: QMS measuring QoS by area

Figure 5.1.4.3-1 depicts a scenario in which the QMS help switch route when some trouble happens in one cell:

1) The V2X application could invoke the QMS API to create several tasks to measure and monitor the QoS of areas associated with potential routes.

2) The QMS producer receives area information provided in step1 and may leverage Location API described in ETSI GS MEC 013 [i.5] to get the measuring targets in Cell2.

3) The QMS measures the targets in Cell2 and evaluate the network performance of Cell2.

4) When there is a QoS degradation in Cell2, the QMS could notify the V2X application that the quality of service cannot satisfy the requirement of V2X service.

5) The V2X application receives the notification from the QMS, and decides to switch the red car to route2.

NOTE: The VIS described in ETSI GS MEC 030 [i.6] provides a similar service to get predicted QoS for a vehicular UE with potential routes. The QMS function described in this clause focuses on evaluating the specific area QoS based on currently active users accessing the target service. The QMS could be regarded as one possible solution to realize the VIS for getting predicted QoS.

5.1.4.4 Measuring QoS for the application undeployed at MEC system

The QMS could help the consumers to measure the QoS regarding to the network on which the application instance works as a server. After instantiating the application, the consumer could leverage QMS to monitor the traffic related to the application instance and get the notification if the network performance degrades. But in some scenarios, the consumer may want to measure or evaluate the network performance before the application instantiation.

For a delay sensitive application, the application provider may want to know if the performance of the edge network could satisfy the application's requirement. In order to obtain an accurate result, the application provider has to deploy the application at different MEC hosts (Generally, the candidates are multiple and the application provider need to compare the performances to make a final decision) to start a series of tests. Unfortunately, application deployment normally is accompanied by complicated and lengthy business negotiations with the operators. And after the tests, if the network metrics, such as latency, packet loss ratio, are not qualified for the application, the application provider has to remove the application instance and application package.

In another scenario, the application server and the application client are still developed in laboratory environment, and the application provider may need to find a way to generate the data flows that could be used to simulate the network performance in real world. Leveraging these data flows, the application provider could adjust software parameters (e.g. minimum buffer, congestion timeout interval etc.) to improve the algorithm for better performance. For example, different congestion control algorithms adapt to different network models. If the network congestion is severe, the application may need to endure a longer congestion timeout. Another example is that the application provider may need to design a larger buffer if the flow data burst, which is related to packet length and packet interval, is common.

NOTE: How to measure the QoS of the edge network when the application is not deployed at MEC system is not specified in the present document.

5.2 Sequence diagrams

5.2.1 General

The following clauses describe how the service consumers interact with the QoS Measurement Service over QMS API to obtain the current QoS and/or predictive QoS. The related sequence diagrams are presented.

5.2.2 QoS measurement subscription

The QoS measurement subscription is the procedure for the QMS consumers to request to receive notifications about QoS measurement. The measurement may be one-off for an immediate query, or periodical for a long time. The QMS consumers could provide a set of terminal device IDs and/or a set of flow filters as the parameters for the QoS measurement task. The QMS consumers could also provide area information and/or available time windows to control the scope of the QoS measurement.

The QMS consumers could leverage the QMS to monitor the quality of specific traffics to get notifications related to QoS event. In this mode, the QMS consumers could set one or more conditions for reporting QoS events, e.g. reporting when the network latency is larger than 20 ms.

Figure 5.2.2-1 shows a flow for QoS measurement subscription.

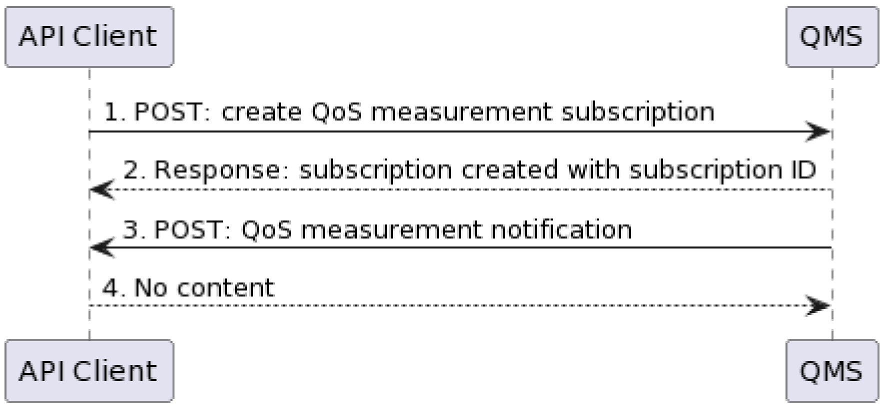
API Client
QMS
1. POST: create QoS measurement subscription
2. Response: subscription created with subscription ID
3. POST: QoS measurement notification
4. No content
API Client
QMS

Figure 5.2.2-1: Flow of QoS measurement subscription

1) The QMS consumer (API Client) subscribes to QoS measurement by sending a request to create a resource containing the subscription details.

2) The QMS returns a response with a resource URI containing the subscriptionId.

3) The QMS reports the up-to-data subscribed information to the QMS consumer by sending a message with the message body containing QoS measurement notification to the callbackURL, which includes QoS measurement, e.g. latency, jitter, etc.

4) The QMS consumer returns a response with the code 204.

5.2.3 Subscription cancellation

The Subscribe Cancellation is the procedure for QMS consumers to cancel the subscription, with which the QMS stop reporting the subscribed information to the QMS consumer.

Figure 5.2.3-1 shows a flow for Subscribe cancellation.

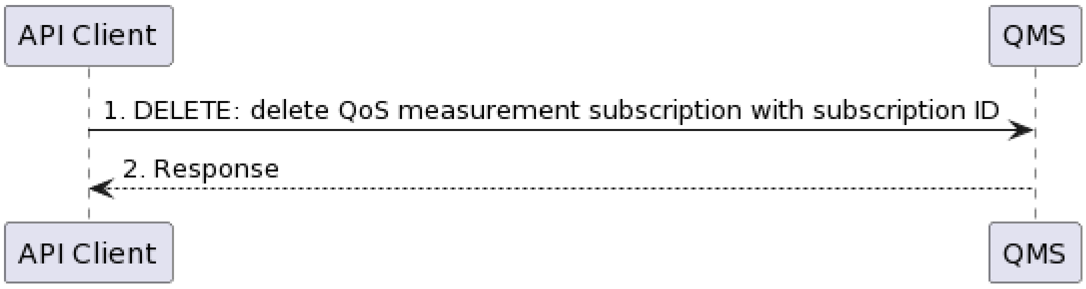
API Client
QMS
1. DELETE: delete QoS measurement subscription with subscription ID
2. Response
API Client
QMS

Figure 5.2.3-1: Flow of subscription cancellation

1) The QMS consumer (API Client) unsubscribes to notifications by sending a request to delete the resource URI containing the subscriptionId.

2) The QMS returns a successful response if the subscription cancellation is accepted.

5.2.4 Getting all QoS measurement subscription

Figure 5.2.4-1 shows a scenario where the API Client (service consumer) sends a request to query all the subscription information related to QoS measurement.

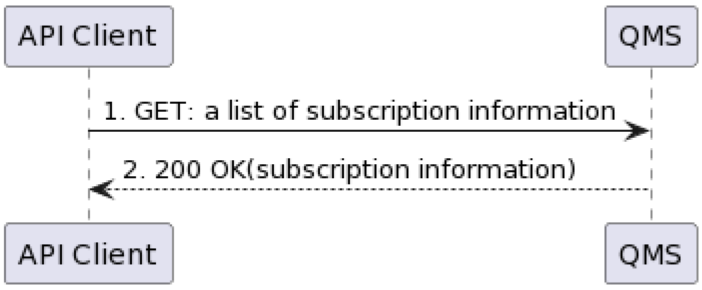
API Client
QMS
1. GET: a list of subscription information
2. 200 OK(subscription information)
API Client
QMS

Figure 5.2.4-1: Flow of Getting QoS measurement subscription

1) API Client (service consumer) sends a request to get all the subscription information related to QoS measurement.

2) QMS returns a response with "200 OK" with the message body containing the list of subscription information.

5.2.5 Getting a specific QoS measurement subscription

Figure 5.2.5-1 shows a scenario where the API Client (service consumer) sends a request to query specific subscription information by subscriptionId.

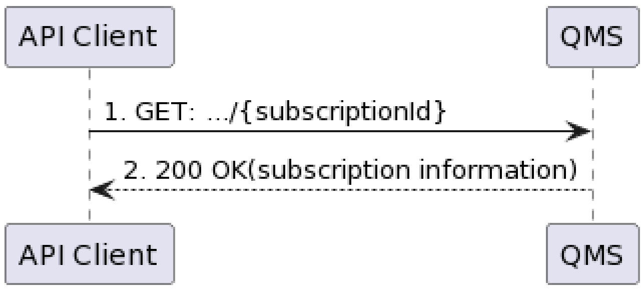
API Client
QMS
1. GET: .../{subscriptionId}
2. 200 OK(subscription information)
API Client
QMS

Figure 5.2.5-1: Flow of Getting a specific QoS measurement subscription

1) API Client (service consumer) sends a request to get specific subscription information by subscriptionId.

2) QMS returns a response with "200 OK" with the message body containing the specific subscription information.

5.2.6 Updating a specific QoS measurement subscription

Figure 5.2.6-1 shows a scenario where the API Client (service consumer) sends a request to update specific subscription information by subscriptionId. A new subscription structure should be carried in the request message body to replace the old subscription information.

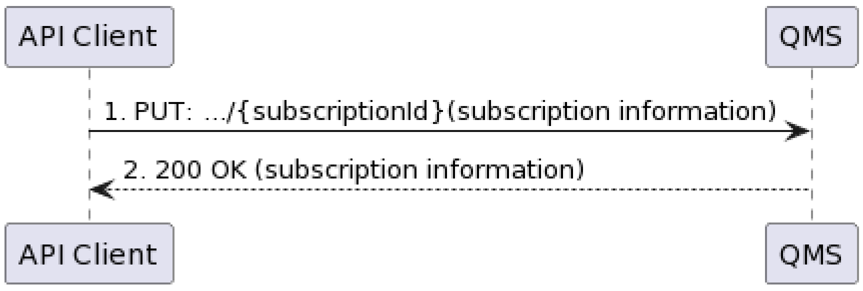
API Client
QMS
1. PUT: .../{subscriptionId}(subscription information)
2. 200 OK (subscription information)
API Client
QMS

Figure 5.2.6-1: Flow of Updating a specific QoS measurement subscription

1) API Client (service consumer) sends a request to update specific subscription information by subscriptionId.

2) QMS returns a response with "200 OK" with the message body containing the updated subscription information.

6 Data model

6.1 Introduction

The following clauses provide the description of the data model.

6.2 Resource data types

Void.

6.3 Subscription data types

6.3.1 Type: NotificationSubscriptionList

This type contains a list of subscriptions.

Table 6.3.1-1: Definition of type NotificationSubscriptionList

Attribute name	Data type	Cardinality	Description
subscription	Structure (inlined)	0..N	
>href	Uri	1	The URI referring to the subscription.
>subscriptionType	String	1	Type of the subscription. The string shall be set according to the "subscriptionType" attribute of the associated subscription data type defined in clauses 6.3.2 and 6.3.3: • "QoSMeasureSubscription". • "QoSEventSubscription".
resourceURL	LinkType	1	Self-referring URL.

6.3.2 Type: QoSMeasureSubscription

This type represents a subscription to the notifications from QMS about QoS measurement.

The attributes of the QoSMeasureSubscription shall follow the indications provided in table 6.3.2-1.

Table 6.3.2-1: Definition of type QoSMeasureSubscription

Attribute name	Data type	Cardinality	Description
subscriptionType	String	1	Shall be set to "QoSMeasureSubscription".
callbackReference	Uri	0..1	URI exposed by the client on which to receive notifications via HTTP. See note 1.
requestTestNotification	Boolean	0..1	Set to TRUE by the service consumer to request a test notification via HTTP on the callbackReference URI, as specified in ETSI GS MEC 009 [2], clause 6.12a.
websockNotifConfig	WebsockNotifConfig	0..1	Provides details to negotiate and signal the use of a Websocket connection between the location server and the service consumer for notifications. See note 1.
_links	Structure (inlined)	0..1	Hyperlink related to the resource. This shall be only included in the HTTP responses and in HTTP PUT requests.
>self	LinkType	1	Self-referring URI. The URI shall be unique within the QoS measurement Subscribe as it acts as an ID for the subscription.
users	array(Uri)	0..N	Address of user (e.g. 'sip' URI, 'tel' URI, 'acr' URI). See note 2.
flowInfo	array(Structure(inlined))	0..N	The information of the measured flows. See note 2.
>samplingRate	Uint32	0..1	The sampling rate determines the proportion of flows that meet the flowFilter to be measured. If not present, all the flows that meet the flowFilter need to be measured. The value should be an integer from 1 to 100. If half of the qualified flows need to be measured, the value should be set to 50.
>flowFilter	Structure(inlined)	1	Traffic flow filtering criteria. If the flowFilter field is included, at least one of its subfields shall be included. Any flowFilter subfield that is not included shall be ignored in traffic flow filtering.
>>sourceIp	String	0..1	Source address identity of measured flow (including range).
>>sourcePort	array(Uint32)	0..N	Source port identity of measured flow.
>>dstIp	String	0..1	Destination address identity of measured flow (including range).
>>dstPort	array(Uint32)	0..N	Destination port identity of measured flow.
>>protocol	Uint32	0..1	Protocol number.

Attribute name	Data type	Cardinality	Description
>>dscp	Uint32	0..1	DSCP in the IPv4 header or Traffic Class in the IPv6 header.
>>flowlabel	Uint32	0..1	Flow Label in the IPv6 header, applicable only if the flow is IPv6.
numberOfReports	Uint32	0..1	If not present, the subscription is active until it is actively terminated via HTTP DELETE. If present, it determines the number of reports to be sent until the subscription gets automatically terminated.
reportingInterval	Uint32	0..1	The reportingInterval determines the interval between two contiguous reports. The unit is second.
measuringPeriod	Uint32	1	It determines measuring frequency. The measuringPeriod shall be less than or equal to reportingInterval. The unit is second.
measuringArea	array(Structure(inlined))	0..N	Area constraint for the measuring task. Traffic of users entering the area will be measured.
>areaInfo	AreaInfo	0..1	It describes the measured area. It shall be present if accessPointId and zoneId are not present.
>accessPointId	String	0..1	The identity of the access point the user is currently on, see note 3.
>zoneId	String	0..1	The identity of the zone the user is currently within, see note 3.
measuringTime	array(Structure(inlined))	0..N	Time constraint for the measuring task. If present, the monitoring task is working only at this specific time section. The time section of [startTime, endTime] may across the midnight.
>startTime	String	1	The format is a string representing the hour, and the minute in a day, like "14:30".
>endTime	String	1	The format is a string representing the hour, and the minute in a day, like "14:30".
metricType	array(MetricType)	1..N	The expected measuring result type.
expiryDeadline	TimeStamp	0..1	The expiration time of the subscription determined by the QoS Measurement Subscribe Service.
NOTE 1: At least one of callbackReference and websockNotifConfig shall be provided by the service consumer. If both are provided, it is up to location server to select an alternative and return only that alternative in the response, as specified in ETSI GS MEC 009 [2], clause 6.12a. NOTE 2: At least one of users and flowInfo shall be provided by the service consumer. NOTE 3: As specified in [3], clause 5.2.2.7.			

6.3.3 Type: QoSEventSubscription

This type represents a subscription to the notifications from QMS about QoS events, e.g. some metrics regarding network performance has achieved the alarm threshold.

The attributes of the QoSEventSubscription shall follow the indications provided in table 6.3.3-1.

Table 6.3.3-1: Definition of type QoSEventSubscription

Attribute name	Data type	Cardinality	Description
subscriptionType	String	1	Shall be set to "QoSEventSubscription".
callbackReference	Uri	0..1	URI exposed by the client on which to receive notifications via HTTP. See note 1.
requestTestNotification	Boolean	0..1	Set to TRUE by the service consumer to request a test notification via HTTP on the callbackReference URI, as specified in ETSI GS MEC 009 [2], clause 6.12a.

Attribute name	Data type	Cardinality	Description
websockNotifConfig	WebsockNotifConfig	0..1	Provides details to negotiate and signal the use of a Websocket connection between the location server and the service consumer for notifications. See note 1.
_links	Structure (inlined)	0..1	Hyperlink related to the resource. This shall be only included in the HTTP responses and in HTTP PUT requests.
>self	LinkType	1	Self-referring URI. The URI shall be unique within the QoS measurement Subscribe as it acts as an ID for the subscription.
users	array(Uri)	0..N	Address of user (e.g. 'sip' URI, 'tel' URI, 'acr' URI). See note 2.
flowFilter	array(Structure(inlined))	0..N	Traffic flow filtering criteria. If the flowFilter field is included, at least one of its subfields shall be included. Any flowFilter subfield that is not included shall be ignored in traffic flow filtering. See note 2.
>sourceIp	String	0..1	Source address identity of measured flow (including range).
>sourcePort	array(Uint32)	0..N	Source port identity of measured flow
>dstIp	String	0..1	Destination address identity of measured flow (including range).
>dstPort	array(Uint32)	0..N	Destination port identity of measured flow.
>protocol	Uint32	0..1	Protocol number.
>dscp	Uint32	0..1	DSCP in the IPv4 header or Traffic Class in the IPv6 header.
>flowlabel	Uint32	0..1	Flow Label in the IPv6 header, applicable only if the flow is IPv6.
reportTrigger	array(Structure(inlined))	1..N	The trigger leading to the notification.
>metricType	MetricType	1	The monitoring metric type.
>upperThreshold	Uint32	0..1	Threshold which if crossed upward shall cause a notification.
>lowerThreshold	Uint32	0..1	Threshold which if crossed downward shall cause a notification.
reportingCtrl	ReportingCtrl	0..1	Provides parameters that ctrl the reporting.
measuringPeriod	Uint32	1	It determines measuring frequency. The unit is second.
monitoringArea	array(Structure(inlined))	0..N	Area constraint for the monitoring task.
>areaInfo	AreaInfo	0..1	It describes the measured area. It shall be present if accessPointId and zoneId are not present.
>accessPointId	String	0..1	The identity of the access point the user is currently on, see note 3.
>zoneId	String	0..1	The identity of the zone the user is currently within, see note 3.
monitoringTime	array(Structure(inlined))	0..N	Time in a day for the monitoring task. If present, the monitoring task is working only at this specific time section. The time section of [startTime, endTime] may across the midnight.
>startTime	String	1	The format is a string representing the hour, and the minute in a day, like "14:30".
>endTime	String	1	The format is a string representing the hour, and the minute in a day, like "14:30".
expiryDeadline	TimeStamp	0..1	The expiration time of the subscription determined by the QoS Measurement Subscribe Service.
NOTE 1: At least one of callbackReference and websockNotifConfig shall be provided by the service consumer. If both are provided, it is up to location server to select an alternative and return only that alternative in the response, as specified in ETSI GS MEC 009 [2], clause 6.12a. NOTE 2: At least one of users and flowFilter shall be provided by the service consumer. NOTE 3: As specified in [3], clause 5.2.2.7.			

6.4 Notifications data types

6.4.1 Type: TestNotification

This type represents a test notification from a location server to determine if the Websocket method is to be utilized for the location server to issue notifications for a subscription, as defined in clause 6.12a of ETSI GS MEC 009 [2].

Table 6.4.1-1: Attributes of the TestNotification

Attribute name	Data type	Cardinality	Description
notificationType	String	1	Shall be set to "TestNotification".
_links	Structure (inlined)	1	Hyperlink related to the resource.
>subscription	LinkType	1	URI identifying the subscription for the test notification.

6.4.2 Type: QoSMeasureNotification

This type represents a notification from QMS with regards to QoS measurement.

The attributes of the QoSMeasureNotification shall follow the indications provided in table 6.4.2-1.

Table 6.4.2-1: Attributes of the QoSMeasureNotification

Attribute name	Data type	Cardinality	Description
notificationType	String	1	Shall be set to "QoSMeasureNotification".
timeStamp	TimeStamp	0..1	Time stamp.
subscriptionState	Enum(inlined)	0..1	It shall be absent if the related subscription has no numberOfReports attribute, and shall be present otherwise. ACTIVE: the subscription is active. FINISHED: This is the last report and the subscription is subject to automatic termination.
qoSMeasureResult	array(Structure(inlined))	0..N	The QoS measuring result.
>user	Uri	0..1	Address of user (e.g. 'sip' URI, 'tel' URI, 'acr' URI). Present if user has been provided in subscription.
>flow	Structure(inlined)	1	Flow information.
>>sourceIp	String	1	Source address identity.
>>sourcePort	Uint32	1	Source port identity.
>>dstIp	String	1	Destination address identity.
>>dstPort	Uint32	1	Destination port identity.
>>protocol	Uint32	0..1	Protocol number.
>>dscp	Uint32	0..1	DSCP in the IPv4 header or Traffic Class in the IPv6 header.
>>flowlabel	Uint32	0..1	Flow Label in the IPv6 header, applicable only if the flow is IPv6.
>measuringArea	Structure(inlined)	0..1	The area information for measurement reporting.
>>areaInfo	AreaInfo	0..1	It shall be present if accessPointId and zoneId are not present.
>>accessPointId	String	0..1	The identity of the access point which the user is currently on, see note 2.
>>zoneId	String	0..1	The identity of the zone which the user is currently within, see note 2.
>measuringTime	Structure(inlined)	0..1	The time section for measurement reporting.
>>startTime	TimeStamp	1	Start time for measurement.
>>endTime	TimeStamp	1	End time for measurement.
>latency	Uint32	0..1	See note 1. The Unit is millisecond.
>jitter	Uint32	0..1	See note 1. The Unit is millisecond.
>throughput	Uint64	0..1	See note 1. The Unit is kbit/s.
>loss_rate	Uint32	0..1	See note 1. The integer represents percent.
>error_rate	Uint32	0..1	See note 1. The integer represents percent.
_links	Structure(inlined)	1	Object containing hyperlinks related to the resource.
>subscription	LinkType	1	A link to the related subscription.

Attribute name	Data type	Cardinality	Description
NOTE 1: The attributes of latency, jitter, throughput, loss_rate, and error_rate should be present according to corresponding subscription. NOTE 2: As specified in [3], clause 5.2.2.7.			

6.4.3 Type: QoSEventNotification

This type represents a notification from QMS with regards to QoS event.

The attributes of the QoSEventNotification shall follow the indications provided in table 6.4.3-1.

Table 6.4.3-1: Attributes of the QoSEventNotification

Attribute name	Data type	Cardinality	Description
notificationType	String	1	Shall be set to "QoSEventNotification".
timeStamp	TimeStamp	0..1	Time stamp.
subscriptionState	Enum(inlined)	0..1	It shall be absent if the related subscription has no numberOfReports attribute, and shall be present otherwise. ACTIVE: The subscription is active. FINISHED: This is the last report and the subscription is subject to automatic termination.
user	Uri	0..1	Address of user (e.g. 'sip' URI, 'tel' URI, 'acr' URI). Present if user has been provided in subscription.
flow	Structure(inlined)	1	Flow information.
>sourceIp	String	1	Source address identity.
>sourcePort	Uint32	1	Source port identity.
>dstIp	String	1	Destination address identity.
>dstPort	Uint32	1	Destination port identity.
>protocol	Uint32	0..1	Protocol number.
>dscp	Uint32	0..1	DSCP in the IPv4 header or Traffic Class in the IPv6 header.
>flowlabel	Uint32	0..1	Flow Label in the IPv6 header, applicable only if the flow is IPv6.
metricType	MetricType	1	The monitoring metric type.
qosEvent	QoSEvent	1	The reporting event.
measuringArea	Structure(inlined)	0..1	The area information for event reporting.
>areaInfo	AreaInfo	0..1	It shall be present if accessPointId and zoneId are not present.
>accessPointId	String	0..1	The identity of the access point which the user is currently on, see note 2.
>zoneId	String	0..1	The identity of the zone which the user is currently within, see note 2.
_links	Structure(inlined)	1	Object containing hyperlinks related to the resource.
>subscription	LinkType	1	A link to the related subscription.
NOTE 1: The attributes of latency, jitter, throughput, loss_rate, and error_rate should be present according to corresponding subscription. NOTE 2: As specified in [3], clause 5.2.2.7.			

6.5 Referenced structured data types

6.5.1 Introduction

This clause defines data structures that are referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any pub/sub mechanism.

6.5.2 Type: WebsockNotifConfig

This type represents configuration for the delivery of subscription notifications over Websockets per the pattern defined in clause 6.12a of ETSI GS MEC 009 [2].

Table 6.5.2-1: Attributes of the WebsockNotifConfig

Attribute name	Data type	Cardinality	Description
websocketUri	Uri	0..1	Set by location server to indicate to the service consumer the Websocket URI to be used for delivering notifications.
requestWebsocketUri	Boolean	0..1	Set to true by the service consumer to indicate that Websocket delivery is requested.

6.5.3 Type: AreaInfo

This type represents the parameters that describe an area.

Table 6.5.3-1: Attributes of type AreaInfo

Attribute name	Data type	Cardinality	Description
shape	Enum(inlined)	1	The shape of the area monitored: 1 = CIRCLE. 2 = POLYGON.
points	array(Point)	1..N	Shall include one point if the shape is CIRCLE. Shall include 3-15 points if the shape is POLYGON.
radius	UnsignedInt	0..1	Shall be present if the shape is CIRCLE.

6.5.4 Type: Point

This type represents the geographical location of a point.

Table 6.5.4-1: Attributes of type Point

Attribute name	Data type	Cardinality	Description
latitude	Float	1	Location latitude, expressed in the range -90° to +90°.
longitude	Float	1	Location longitude, expressed in the range -180° to +180°.

6.5.5 Type: TimeStamp

This type represents a time stamp.

Table 6.5.5-1: Attributes of type TimeStamp

Attribute name	Data type	Cardinality	Description
seconds	Uint32	1	The seconds part of the time. Time is defined as Unix-time since January 1, 1970, 00:00:00 UTC.
nanoSeconds	Uint32	1	The nanoseconds part of the time. Time is defined as Unix-time since January 1, 1970, 00:00:00 UTC.

6.5.6 Type: ReportingCtrl

This type represents the parameters that control the report times and frequency.

Table 6.5.6-1: Attributes of type ReportingCtrl

Attribute name	Data type	Cardinality	Description
minimumInterval	UnsignedInt	0..1	Minimum interval between reports in case frequently reporting. Unit is second.
maximumFrequency	UnsignedInt	0..1	Maximum frequency (in seconds) of notifications per subscription.
maximumCount	UnsignedInt	0..1	Maximum number of notifications. For no maximum, either do not include this element or specify a value of zero. Default value is 0.

6.6 Referenced simple data types and enumerations

6.6.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

6.6.2 Type: LinkType

This type represents a type of link and may be referenced from data structures.

Table 6.6.2-1: Definition of type LinkType

Attribute name	Data type	Cardinality	Description
href	Uri	1	URI referring to a resource.

6.6.3 Enumeration: MetricType

This type represents QoS measuring result type.

Table 6.6.3-1: Enumeration MetricType

Type name	Description
"LATENCY"	The latency of the measured traffic.
"JITTER"	The jitter of the measured traffic.
"THROUGHPUT"	The throughput of the measured traffic.
"LOSS_RATE"	The packet loss rate of the measured traffic.
"ERROR_RATE"	The packet error rate of the measured traffic.

6.6.4 Enumeration: QoSEvent

This type represents QoS reporting event type.

Table 6.6.4-1: Enumeration QoSEvent

Type name	Description
"ABOVE_UPPER_THRESHOLD"	The event represents the value is above the upper threshold.
"BELOW_LOWER_THRESHOLD"	The event represents the value is below the lower threshold.

7 API definition

7.1 Introduction

This clause defines the resources and operations of the QoS Measurement Service (QMS) API.

7.2 Global definitions and resource structure

All resource URIs of this API shall have the following root:

{apiRoot}/{apiName}/{apiVersion}/

Where:

- The "apiRoot" consists of the scheme ("https"), host and optional port, and an optional prefix string. It can be discovered using the service registry.
- The "apiName" shall be set to "qms".
- The "apiVersion" shall be set to "v1" for the present document. All resource URIs in the sub-clauses below is defined relative to the above root URI.

The API shall support HTTP over TLS as defined in clause 6.22 of ETSI GS MEC 009 [2].

This API shall use OAuth 2.0, as defined in clause 6.16 of ETSI GS MEC 009 [2]. This OAuth 2.0 authorization procedure shall occur only on TLS-protected connections.

This API supports additional application-related error information to be provided in the HTTP response when an error occurs. See clause 6.15 of ETSI GS MEC 009 [2] for more information.

The content format JSON shall be supported.

The JSON format shall be signalled by the content type "application/json".

Figure 7.2-1 illustrates the resource URI structure of this API. Table 7.2-1 provides an overview of the resources defined by the present document, and the applicable HTTP methods.

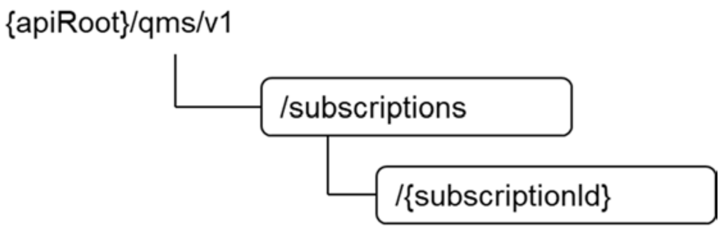

Figure 7.2-1: Resource URI structure of the QMS API

Table 7.2-1: Resources and methods overview

Resource name	Resource URI	HTTP method	Meaning
A list of subscriptions	/subscriptions	GET	Retrieve information about a list of subscriptions.
		POST	Create an individual subscription.
Individual subscription	/subscriptions/{subscriptionId}	GET	Retrieve information about a specific subscription.
		PUT	Modify the information about a specific subscription.
		DELETE	Delete a specific subscription.

7.3 Resource: a list of subscriptions

7.3.1 Description

This resource is used to represent a list of subscriptions.

7.3.2 Resource definition

Resource URI: **{apiRoot}/subscriptions**

This resource shall support the resource URI variables defined in table 7.3.2-1.

Table 7.3.2-1: Resource URI variables for resource "a list of subscriptions"

Name	Definition
apiRoot	See clause 7.2.

7.3.3 Resource methods

7.3.3.1 GET

The GET method is used to retrieve information about a list of subscriptions.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the tables 7.3.3.1-1 and 7.3.3.1-2.

Table 7.3.3.1-1: URI query parameters supported by the GET method on this resource

Name	Data type	Cardinality	Remarks
subscriptionId	String	0..N	Multiple subscriptionId may be used as an input parameter to query a list of subscriptions.
subscriptionType	String	0..1	subscriptionType may be used as an input parameter to query the availability of a list of subscriptions.

Table 7.3.3.1-2: Data structures supported by the GET request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response Codes	Remarks
	NotificationSubscriptionList	1	200 OK	Upon success, a response body containing the list of links to requestor's subscriptions is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
	ProblemDetails	0..1	414 URI Too Long	It is used to indicate that the server is refusing to process the request because the request URI is longer than the server is willing or able to process.

7.3.3.2 PUT

Not supported.

7.3.3.3 PATCH

Not supported.

7.3.3.4 POST

The POST method is used to create an individual subscription.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the tables 7.3.3.4-1 and 7.3.3.4-2.

Table 7.3.3.4-1: URI query parameters supported by the POST method on this resource

Name	Data type	Cardinality	Remarks
n/a			

Table 7.3.3.4-2: Data structures supported by the POST request/response on this resource

	Data type	Cardinality	Response Codes	Remarks
Request body	{NotificationSubscription}	1		The entity body in the request contains data type of QoS measurement subscription that is to be created, where the data type options are listed below and defined in clauses 6.3.2 and 6.3.3: • QoSMeasureSubscription. • QoSEventSubscription.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	{NotificationSubscription}	1	201 Created	Indicates successful resource creation, where the resource URI shall be returned in the HTTP Location header field. In the returned NotificationSubscription structure, the created subscription is described using the appropriate data type from the list below and as defined in clauses 6.3.2 and 6.3.3: • QoSMeasureSubscription. • QoSEventSubscription.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

7.3.3.5 DELETE

Not supported.

7.4 Resource: individual subscription

7.4.1 Description

This resource is used to represent a subscription.

7.4.2 Resource definition

Resource URI: {apiRoot}/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 7.4.2-1.

Table 7.4.2-1: Resource URI variables for resource individual subscription

Name	Definition
apiRoot	See clause 7.2.
subscriptionId	Represents a subscription.

7.4.3 Resource methods

7.4.3.1 GET

The GET method is used to retrieve information about an individual subscription.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the tables 7.4.3.1-1 and 7.4.3.1-2.

Table 7.4.3.1-1: URI query parameters supported by the GET method on this resource

Name	Data type	Cardinality	Remarks
n/a			

Table 7.4.3.1-2: Data structures supported by the GET request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	{NotificationSubscription}	1	200 OK	Upon success, a response body containing data type describing the QoS measurement subscription is returned. The allowed data types for subscriptions are defined in clauses 6.3.2 and 6.3.3: • QoSMeasureSubscription. • QoSEventSubscription.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

7.4.3.2 PUT

The PUT method is used to modify the information of an individual subscription.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the tables 7.4.3.2-1 and 7.4.3.2-2.

Table 7.4.3.2-1: URI query parameters supported by the PUT method on this resource

Name	Data type	Cardinality	Remarks
n/a			

Table 7.4.3.2-2: Data structures supported by the PUT request/response on this resource

	Data type	Cardinality	Remarks	
Request body	**{NotificationSubscription}**	1	New NotificationSubscription is included as entity body of the request. The allowed data types for subscriptions are defined in clauses 6.3.2 and 6.3.3: • QoSMeasureSubscription. • QoSEventSubscription.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	{NotificationSubscription}	1	200 OK	Upon success, a response body containing data type describing the updated subscription is returned. The allowed data types for subscriptions are defined in clauses 6.3.2 and 6.3.3: • QoSMeasureSubscription. • QoSEventSubscription.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
	ProblemDetails	0..1	412 Precondition Failed	It is used when a condition has failed during conditional requests, e.g. when using ETags to avoid write conflicts. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.

7.4.3.3 PATCH

Not supported.

7.4.3.4 POST

Not supported.

7.4.3.5 DELETE

The DELETE method is used to delete an individual subscription.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the tables 7.4.3.5-1 and 7.4.3.5-2.

Table 7.4.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Data type	Cardinality	Remarks
n/a			

Table 7.4.3.5-2: Data structures supported by the DELETE request/response on this resource

Request body	**Data type**	**Cardinality**	**Remarks**	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	n/a	1	204 No Content	The operation has been successful. The response message content shall be empty.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

Annex A (informative): Complementary material for API utilization

To complement the definitions for each method and resource defined in the interface clauses of the present document, ETSI MEC ISG is providing for the user application lifecycle management API a supplementary description file compliant to the OpenAPI™ Specification [i.8].

In case of discrepancies between the supplementary description file and the related data structure definitions in the present document, the data structure definitions take precedence.

The supplementary description file, relating to the present document, is located at https://forge.etsi.org/rep/mec/gs045-qos-mea-api.

History

Document history		
V3.1.1	March 2024	Publication