UPnP QosManager:3
Service Template Version 1.01

For UPnP Version 1.0
Status: Standardized DCP
Date: November 30, 2008

This Standardized DCP has been adopted as a Standardized DCP by the Steering Committee of the UPnP Forum, pursuant to Section 2.1(c)(ii) of the UPnP Forum Membership Agreement. UPnP Forum Members have rights and licenses defined by Section 3 of the UPnP Forum Membership Agreement to use and reproduce the Standardized DCP in UPnP Compliant Devices. All such use is subject to all of the provisions of the UPnP Forum Membership Agreement.

THE UPNP FORUM TAKES NO POSITION AS TO WHETHER ANY INTELLECTUAL PROPERTY RIGHTS EXIST IN THE STANDARDIZED DCPS. THE STANDARDIZED DCPS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". THE UPNP FORUM MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE STANDARDIZED DCPS, INCLUDING BUT NOT LIMITED TO ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, OF REASONABLE CARE OR WORKMANLIKE EFFORT, OR RESULTS OR OF LACK OF NEGLIGENCE.

Authors	Member
Ally Yu-kyoung Song	LGE
Amol Bhagwat (editor)	CableLabs
Bruce Fairman	Sony
Daryl Hlasny	Sharp Labs of America
Dieter Verslype	Ghent University
Fred Tuck (co-chair)	EchoStar
Jelle Nelis	Ghent University
Michael van Hartskamp (co-chair)	Philips
Narm Gadiraju	Intel Corporation
Puneet Sharma	HP
Richard Chen	Philips
Sherman Gavette	Sharp Labs of America
Steve Wade	Sharp Labs of America
Suman Sharma	Intel Corporation

Authors	Member
Zong Wu	Entropic

The UPnP Forum in no way guarantees the accuracy or completeness of this author list and in no way implies any rights for or support from those members listed. This list is not the specifications' contributor list that is kept on the UPnP Forum's website.

Contents

List of Tables

1. Overview and Scope

This service definition is compliant with the UPnP Device Architecture version 1.0.

This service-type enables modeling of 'Quality of Service Manager' function capabilities. *QosManager* functionality is the combination of *QosManager* service and control point functionality that discovers and controls *QosDevice* and *QosPolicyHolder* services running on the network. The *QosManager* function is responsible for requesting, updating, releasing and in general controlling the Quality of Service assigned by networking devices to various traffic streams. The *QosManager* service is invoked from an UPnP Control Point to perform the functions related to setting up QoS[2] for that traffic stream. Once the network is configured with respect to the QoS for the upcoming traffic stream, the *QosManager* service will hand back control to the Control Point. This service provides a mechanism for a Control Point to:

- Be agnostic of the QoS capabilities and associated details about the various QoS Devices on the network at the expense of potential network inefficiency.
- Abstract the tasks of setting up, modifying and revoking the QoS associated with every traffic stream
- *QosManager:3* defines new capabilities to set up Parameterized QoS. Control Points may now request reservations for specific resources for the exclusive use of individual streams. Successful resource reservation ensures that streams will always have the bandwidth and other resources to transport time critical data.
- *QosManager:3* defines a method for Control Points to request preemption of existing reservations on the UPnP-QoS network if necessary to admit a new stream.
- *QosManager:3* supports a mechanism for providing a list of blocking streams to a Control Point so that it can handle preemption on its own, if desired.

A *QosManager* is a dual-role entity that exposes a *QosManager* Service to the Control Point while acting as a Control Point for the *QosPolicyHolder* Service and *QosDevice* Services running on the network. This document describes the components of the UPnP *QosManager* Service and the *QoS Manager*. The *QoS Manager* provides the Control Point functionality that discovers and controls *QosDevice* Services and the *QosPolicyHolder* Services running on the network. Additional information concerning the *QoS Manager* may be found in:

- UPnP QoS Architecture document [QoS Architecture]
- UPnP QosDevice Service Definition Document [QOS DEVICE]
- UPnP QosPolicyHolder Service Definition Document [POLICY HOLDER]

1.1. References & Terms

1.1.1. References

Unless explicitly stated otherwise herein, implementation of the mandatory provisions of any standard referenced by this specification shall be mandatory for compliance with this specification. This section lists the normative references used in this document and includes the tag inside square brackets that is used for each sub reference:

[2] Quality of Service

1.1.1.1. Normative References

[XML] – *Extensible Markup Language (XML) 1.0 (Second Edition)*, T. Bray, J.Paoli, C. M. Sperberg McQueen, E Maler, eds. W3C Recommendations, 6 October 2000.

[DEVICE] - UPnP Device Architecture, version 1.0, UPnP Forum, July 20, 2006.
Available at: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20060720.pdf
Latest version available at: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf

[POLICY HOLDER] – UPnP QosPolicyHolder:3 Service Document.
Available at: http://www.upnp.org/specs/qos/UPnP-qos-QosPolicyHolder-v3-Service-20081130.pdf
Latest version available at: http://www.upnp.org/specs/qos/UPnP-qos-QosPolicyHolder-v3-Service.pdf

[QOS DEVICE] – UPnP QosDevice:3 Service Document.
Available at: http://www.upnp.org/specs/qos/UPnP-qos-QosDevice-v3-Service-20081130.pdf
Latest version available at: http://www.upnp.org/specs/qos/UPnP-qos-QosDevice-v3-Service.pdf

[QDA:3] – UPnP QosDevice:3 Underlying Technology Interface Addendum
Available at: http://www.upnp.org/specs/qos/UPnP-qos-QosDevice-v3-Addendum-20081130.pdf
Latest version available at: http://www.upnp.org/specs/qos/UPnP-qos-QosDevice-v3-Addendum.pdf

[IANA Portnumbers] – *Port Numbers*, IANA, Available at: http://www.iana.org/assignments/port-numbers

[IANA Protocols] – *Assigned Internet Protocol Numbers*, IANA,
Available at: http://www.iana.org/assignments/protocol-numbers

[RFC 2141] – *URN Syntax*, R. Moats, May 1997, Available at: http://www.ietf.org/rfc/rfc2141.txt

[RFC 2211] –*Specification of the Controlled-Load Network Element Service*, J. Wroclawski, September 1997.
Available at: http://www.ietf.org/rfc/rfc2211.txt

[RFC 2212] – Specification of Guaranteed Quality of Service, S. Shenker et al., September 1997. Available at: http://www.ietf.org/rfc/rfc2212.txt

[RFC 3339] – *Date and Time on the Internet: Timestamps*, G. Klyne, July 2002.
Available at: http://www.ietf.org/rfc/rfc3339.txt

[RFC 3927] – *Dynamic Configuration of IPv4 Link-Local Addresses*, Internet Engineering Task Force, S. Cheshire et al., March 2005.
Available at: http://www.ietf.org/rfc/rfc3927.txt

[RFC 3986] –Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee et al., January 2005
Available at: ftp://ftp.rfc-editor.org/in-notes/rfc3986.txt

1.1.1.2. Informative References

This section lists the informative references used in this document and includes the tag inside square brackets that is used for each sub reference:

[CDS:2] – UPnP AV ContentDirectory Service Definition document version 2.0.
Available at: http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v2-Service.pdf
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v2-Service-20060531.pdf

[IEEE 802.3-2002] – *IEEE Std 802.3, 2000 Edition, Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*, LAN MAN Standards Committee of the IEEE Computer Society, 2000.

[IEEE 802.1D] – *IEEE Std 802.1Q– 2018, Media Access Control (MAC) Bridges*, IEEE Computer Society, 2018.

[QoS Architecture] – *UPnP QoS Architecture V3.0*
Available at: http://www.upnp.org/specs/qos/UPnP-qos-Architecture-v3-20081130.pdf
Latest version available at: http://www.upnp.org/specs/qos/UPnP-qos-Architecture-v3.pdf

[HPAV] – HomePlug AV Specfication v1.1, HomePlug Power Alliance, 2007.

[PACKETC] – PacketCable Specification Multimedia Specification,
Available at: http://www.packetcable.com/downloads/specs/PKT-SP-MM-I03-051221.pdf

[RFC 1363] – A Proposed Flow Specification, C. Partridge, September 1992,
Available at: http://www.ietf.org/rfc/rfc1363.txt

1.1.2. Terms and Abbreviations

1.1.2.1. Abbreviations

Table 1-1:Abbreviations

Abbreviation	Description
AV	Audio/Video
CP	Control Point
DCP	Device Control Protocol
DSCP	Differentiated Services Code Point
IANA	Internet Assigned Numbers Authority
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IETF	Internet Engineering Task Force
IGD	Internet Gateway Device
IP	Internet Protocol
LAN	Local Area Network
MAC	Medium Access Control
PVR	Personal Video Recorder
QD	*QosDevice* Service
QM	*QosManager* Service
QoS	Quality of Service
QPH	*QosPolicyHolder* Service
SRP	Stream Reservation Protocol
STA	Station
STB	Settop Box
TSPEC	Traffic Specification
WMM	Wireless Multi Media
WMM-SA	Wireless Multi Media Scheduled Access

Abbreviation	Description
XML	eXtensible Markup Language

1.1.2.2. Terms

1.1.2.2.1. Admission Control

Admission control is the overall procedure that consists of

- Breaking down (by a *QoS Manager*) of the end-to-end requirements into per *QoS segment* requirements;
- Requesting the reservation of resources (at the *QosDevice* Service) and the process that is carried out by a *QosDevice* Service to reserve those resources;
- Applying the results from the *QosPolicyHolder* Service to determine the priority and / or importance of the stream and whether it should be admitted at the possible expense of other streams.

The overall result is a yes/no decision of whether the traffic stream can be supported.

1.1.2.2.2. Admission Mechanism

Admission mechanism is the implementation of the resource management scheme of a *QosDevice* Service. The *QosDevice* Service abstracts the properties of a particular technology and the UPnP-QoS interface provides access to the *Admission Mechanism* via the defined actions. The resource management scheme may operate at any layer of the ISO stack, although it frequently uses L2 signaling.

1.1.2.2.3. Admission Policy

Admission policy is the set of rules that is applied in an *admission control* procedure. The rules are applied on the parameters from the *TrafficPolicy* as supplied by the *QosPolicyHolder* Service and the Traffic Specification.

1.1.2.2.4. AV Control Point

A *UPnP Control Point* that is used in an AV scenario and utilizes UPnP AV services is called an *AV Control Point*.

1.1.2.2.5. Best Effort

Best effort is a service of the Internet Protocol; i.e. it makes almost no guarantees about a packet other than attempting to transmit it. At the destination, the packet may arrive damaged, it may be out of order (compared to other packets sent between the same hosts), it may be duplicated, or it may be dropped entirely. If reliability is needed, it is provided by upper level protocols transported using IP.

With the introduction of IEEE 802.1D *priority tagging* [IEEE 802.1D] the priority associated with untagged packets is also called "Best Effort".

1.1.2.2.6. Blocked Segments

A *segment* is called *blocked* for the setup of a new traffic stream, if the *admission* on the segment failed due to insufficient *resources* somewhere in the segment.

1.1.2.2.7. Blocking Traffic Stream

An *admitted traffic stream* is called *blocking* with respect to a new traffic stream S, if it uses network resources where the admission of S fails.

1.1.2.2.7.1. Bridge

A *device* that interconnects two or more LANs that use the same Data Link layer protocols above the MAC sublayer, but can use different MAC protocols.

1.1.2.2.8. Complex Type

A *complex type* is defined in the XML Schema Definition Language [XML] .

1.1.2.2.9. Container

A *container* is a UPnP-QoS construct that contains *containers* and/or *fields*. In XML it is expressed through elements of *complex type*.

1.1.2.2.10. Device

A *device* is a functional unit for a particular purpose. A device is not necessarily a *UPnP Device*.

1.1.2.2.11. Element

An *element* is defined in the XML Schema Definition Language [XML] .

1.1.2.2.12. End-to-End Delay

The End-to-End delay of an application payload is defined as the time difference between the delivery of the payload for sending by the application on the source device, and the delivery of the payload to the destination device's application. The time reference used at the source and destination is the same and is known at the application layer. The End-to-End delay experienced by the packets of a stream is not constant and the delay will have a maximum and a minimum, for a period of observation. An application may, for example, require that the maximum End-to-End delay be less than or equal to an actual value it desires. See [QOS DEVICE] for more details.

1.1.2.2.13. End-to-End Jitter

End-to-End Jitter is defined as the difference between the maximum of End-to-End Delay and the minimum of End-to-End Delay.

1.1.2.2.14. Existing User

When pre-empting an existing traffic stream, the user(s) enjoying this stream is/are affected. In a pre-emption discussion, this user is called the "*Existing User*".

1.1.2.2.15. Field

A *field* is a UPnP-QoS construct that provides information on properties, capabilities, etc. In XML it is expressed through an element of simple type.

1.1.2.2.16.Hub

A device, with two or more physical ports, that forwards all traffic received on any individual port to all other ports. This device is also referred to as a "repeater". In a home network, a hub connects networks segments of the same physical medium.

1.1.2.2.17.Interface

The point of interconnection between a device and a network**.**

1.1.2.2.18.Intermediate Device

An *intermediate device* is physically connected between the Source and Sink device traffic stream. There may be zero or more intermediate device between the Source to Sink.

1.1.2.2.19.Internet Gateway Device (IGD)

Internet Gateway Device is a border device that physically connects the Home Network with a Wide Area Network. This device performs routing. QoS mechanisms associated with routing are not addressed by this architecture; the IGD may present a *QosDevice* Service on the Home Network.

1.1.2.2.20.L2 Technology

The term *L2 Technology* is short hand to refer to MAC / PHYs of various LAN technologies. Examples include: IEEE 802.3, MOCA, HomePlug.

1.1.2.2.21.Layer 2 Stream Id

A Layer 2 construct to uniquely identify a traffic stream within a *QoS segment*.

1.1.2.2.22.Lease Time

The *lease time* is the finite duration for which the resources are reserved for a traffic stream.

1.1.2.2.23.Link

A *link* is an (optionally bidirectional) direct connection between two devices for data exchange. This is called a link segment in the terminology of [IEEE 802.3-2002] .

1.1.2.2.24.Path

A *path* is the physical course that traffic stream will follow from source to sink. For UPnP-QoS, a path must reside within a single IP subnet, but a path may comprise multiple segments. This is called a link in the terminology of [IEEE 802.3-2002] .

1.1.2.2.25.Path Determination

Path determination is the process of determining the ordered set of *QosDevice* Service(s) on the path of a given traffic stream.

1.1.2.2.26.Packet Priority

The *Packet Priority* is a Layer 2 parameter that indicates the priority handling requested by the source. Typically this parameter is part of a packet header and indicates the relative importance of the packet compared to other

packets. It is used to differentiate packets to determine which are given preferential access to the communication medium. This parameter is typically mapped to an Access Priority value supported by the network device.

1.1.2.2.27.Parameterized QoS

Parameterized QoS refers to a general methodology for obtaining QoS. A contract is made with the network using a set of parameters that define the application's traffic requirements. The parameters typically include bandwidth and delay.

1.1.2.2.28.Policing

Policing is the process of ensuring that no more than the reserved resources are used.

1.1.2.2.29.Policy

Policy is a set of rules that is applied for the allocation of resources.

1.1.2.2.30.Policy Management

Policy Management is a function that applies policies to traffic streams.

1.1.2.2.31.Preempted Traffic Stream

A *preempted traffic stream* is a blocking traffic stream that has its reservation modified or deleted by the *QoS Manager*.

1.1.2.2.32.Preemption

When an admission request fails due to the lack of resources, the *QoS Manager* may attempt to free resources necessary to allow the new traffic stream to reserve them. The process of taking resources from existing admitted traffic streams is called *preemption*.

1.1.2.2.33.Preemption Candidate

A *preemption candidate* is a blocking traffic stream that has been selected for possible preemption by the *QoS Manager*.

1.1.2.2.34.Preemption Style

A *preemption style* is a UPnP-QoS defined method of how the *QoS Manager* selects preemption candidates.

1.1.2.2.35.Prioritized QoS

Prioritized QoS refer to a general methodology for providing QoS by differentiating traffic. Messages types are grouped by order of importance and assigned a priority. Message types assigned a higher priority are given preferential access to the communications medium.

1.1.2.2.36.Quality of Service (QoS)

The term *Quality of Service* (abbreviated QoS) refers to a broad collection of networking capabilities and techniques. The goal of QoS is to improve the user experience of a network's ability to deliver predictable results for sensitive applications such as audio, video, and voice applications. Elements of network performance within the scope of QoS often include bandwidth (throughput), latency (delay), and error rate. There are two broad

classes of QoS: data reliability and temporal reliability. Each makes different demands on network technologies. This architecture is primarily concerned with delivering temporal reliability.

1.1.2.2.37.QoS segment

A *QoS segment* is defined as a collection of *interfaces* of *QosDevice* services reporting the same *QosSegmentId* (as defined in [QOS DEVICE]), and the network resources (e.g., wires) through which these interfaces are connected.

1.1.2.2.38. QoS Segment Delay

QoS Segment Delay represents one of the following: 1. The delay at the *QosDevice* including the interfaces; or 2. The delay introduced by all or part of the QoS Segment between the *QosDevice*s including the interfaces. The sum of the QoS Segment Delay values for *QosDevice*s in the path on a QoS segment must be the total delay introduced on that QoS segment. The QoS Segment Delay for a *QosDevice* service is technology and implementation specific. Details on calculating the QoS Segment Delay can be found in the *QosDevice* Service document and the technology specific addendum [QDA:3].

1.1.2.2.39.QoS Segment Jitter

The QoS Segment Jitter is defined as the the difference between the maximum of QoS Segment Delay and the minimum of QoS Segment Delay of the same QoS Segment.

1.1.2.2.40.Resource

In the context of QoS, a *resource* is anything managed in order to support a QoS function. A distinction can be made between device resources and network resources.

1.1.2.2.41.Simple Type

A *simple type* is defined in the XML Schema Definition Language.

1.1.2.2.42.Sink Device

A *sink device* is the receiving end of a traffic stream regardless of other functions on the device.

1.1.2.2.43.Source Device

A *source device* is the originating end of a traffic stream regardless of other functions on the device.

1.1.2.2.44.Subnet

Subnet as defined in [RFC 3927] .

1.1.2.2.45.Switch

A *switch* is a device, with two or more similar or dissimilar physical ports, that sends traffic received on any individual port to the appropriate other port(s) based on layer 2 information.

1.1.2.2.46.Traffic Class

The *Traffic Class* indicates the kind of traffic in the traffic stream. The Traffic Class is used to distinguish, for example, audio from video. The distinction is at the application layer and the Traffic Class is mapped into the

applicable layer 2 representations for the L2 Technology bearing the stream. An example is the mapping in IEEE 802.1D, Annex G [IEEE 802.1D].

1.1.2.2.47.Traffic Descriptor

The *TrafficDescriptor* is the representation of a traffic stream for purposes of providing QoS. It is composed of a TrafficId and other pertinent information that must be provided by the Control Point.

1.1.2.2.48.Traffic Identifier

A *Traffic Identifier* (*TrafficId*) is a set of information that uniquely identifies a data packet as belonging to a traffic stream. This information is typically used by a packet classifier function to associate a Traffic Specification's QoS contract to the service provided to the Traffic stream. Other technologies may refer to this as a Filter Spec (RFC2205) or Traffic Classifier (IEEE 802.11e).

1.1.2.2.49.Traffic Specification (TSPEC)

A *Traffic Specification* (*TSPEC*) contains a set of parameters that define the characteristics of the traffic stream. The TSPEC defines the resource requirements for carrying the traffic stream.

1.1.2.2.50.Traffic Stream

A *Traffic Stream* is a unidirectional flow of data. It originates at a source device and terminates at a sink device.

1.1.2.2.51.UPnP Device

A *UPnP Device* is defined in the UPnP Device Architecture[DEVICE] .

2. Service Modeling Definitions

2.1. ServiceType

The following service type identifies a service that is compliant with this template:

urn:schemas-upnp-org:service:*QosManager:3*

2.2. State Variables

Reader Note: For first-time reader, it may be more insightful to read the theory of operations first and then the action definitions before reading the state variable definitions.

2.2.1. XML Fragments as UPnP Arguments

UPnP-QoS often uses XML Fragments as arguments in UPnP actions. The containing UPnP data type is a **string**. This places restrictions on a string's content; it has to represent a well-formed XML fragment (this includes a complete XML document).

An XML fragment, in adherence to the UPnP V1.0 architecture [DEVICE], MUST be escaped by using the normal XML rules, [XML] Section 2.4 Character Data and Markup, before embedding it in a SOAP request / response message or an event notification message. The XML escaping rules are summarized:

- The (<) character is encoded as (<)
- The (>) character is encoded as (>)
- The (&) character is encoded as (&)
- The (") character is encoded as (")
- The (') character is encoded as (')

In their XML fragments, implementations MAY use an explicit reference to appropriate namespaces.

Table 2-1: State Variables

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value [2]	Default Value [2]	Eng. Units
A_ARG_TYPE_TrafficDescriptor	R	String (XML fragment)	See §2.2.2	n/a	n/a
A_ARG_TYPE_TrafficHandle	R	String (XML fragment)	See §2.2.3	n/a	n/a
A_ARG_TYPE_NumTrafficDescriptors	R	ui4	See §2.2.4	n/a	n/a
A_ARG_TYPE_NumPolicyHolders	R	ui4	See §2.2.5	n/a	n/a

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value [2]	Default Value [2]	Eng. Units
A_ARG_TYPE_ListOfTrafficDescriptors	R	String (XML fragment)	See §2.2.6	n/a	n/a
A_ARG_TYPE_QmCapabilities	R	String (XML fragment)	See §2.2.7	n/a	n/a
A_ARG_TYPE_ExtendedTrafficQosInfo	O	String (XML fragment)	See §2.2.8	n/a	n/a
A_ARG_TYPE_ResultedQosType	R	ui4	See §2.2.9	n/a	n/a

[1] R = Required, O = Optional, X = Non-standard.

[2] Values listed in this column are required. To specify standard optional values or to delegate assignment of values to the vendor, you MUST reference a specific instance of an appropriate table below.

2.2.2. A_ARG_TYPE_TrafficDescriptor

The *A_ARG_TYPE_TrafficDescriptor* state variable contains information about a particular traffic stream.

2.2.2.1. XML Schema Definition

This is a **string** containing an XML fragment. It contains information describing a traffic descriptor. The XML fragment in this argument MUST validate against the XML schema for `TrafficDescriptor` in the XML namespace "http://www.upnp.org/schemas/TrafficDescriptorv1.xsd" which is located at "http://www.upnp.org/schemas/qos/TrafficDescriptor-v3.xsd".

2.2.2.2. Description of fields in the TrafficDescriptor structure

2.2.2.2.1. TrafficHandle

This is an OPTIONAL field in the TrafficDescriptor and is not related to *A_ARG_TYPE_TrafficHandle*. *TrafficHandle* is a unique identifier associated with a particular instance of *TrafficDescriptor*; i.e. a particular traffic stream. It is a unique string generated by the *QosManager* Service and provided to the Control Point in response to the *QM:RequestTrafficQos()* or *QM:RequestExtendedTrafficQos()* action. The *QoS Manager* must ensure that the generated Traffic Handle string is unique for all Traffic Descriptors on the network. In all subsequent communication between the Control Point and the *QosManager* Service, *TrafficHandle* is used to uniquely reference a particular traffic stream.

2.2.2.2.2. TrafficId

This is a MANDATORY structure. *TrafficId* contains information about identifying/classifying packets that belong to that particular traffic stream. It consists of an XML structure consisting of: *SourceAddress*, *DestinationAddress*, *SourcePort*, *DestinationPort*, and *IpProtocol*. The embedded *v3TrafficId* further contains *SourceUuid* and *DestinationUuid*.

2.2.2.2.2.1. SourceAddress

This is an OPTIONAL or MANDATORY field depending on the action. The *SourceAddress* identifies the source of the traffic stream. A value of 0.0.0.0 represents a 'Don't Care' i.e., the condition where a Control Point wants to set up QoS for all traffic streams irrespective of the *SourceAddress*.

2.2.2.2.2.2. DestinationAddress

This is an OPTIONAL or MANDATORY field depending on the action. The *DestinationAddress* identifies the destination of the traffic stream. A value of 0.0.0.0 represents a 'Don't Care' i.e., the condition where a Control Point wants to set up QoS for all traffic streams irrespective of the *DestinationAddress*.

2.2.2.2.2.3. SourcePort

This is an OPTIONAL or MANDATORY field depending on the action. The *SourcePort* identifies the port of the source of the traffic stream. A value of 0 represents a 'Don't Care' i.e., the condition where a Control Point wants to set up QoS for all traffic streams irrespective of the *SourcePort*.

2.2.2.2.2.4. DestinationPort

This is an OPTIONAL or MANDATORY field depending on the action. The *SourcePort* identifes the port of the destination of the traffic stream. A value of 0 represents a 'Don't Care' i.e., the condition where a Control Point wants to set up QoS for all traffic streams irrespective of the *DestinationPort*.

2.2.2.2.2.5. IpProtocol

This is an OPTIONAL or MANDATORY field depending on the action. The value of *IpProtocol* is an IANA assigned IP protocol number from 0-255 (for more information: [IANA Protocols]). A value of 0 represents a 'Don't Care' i.e., the condition where a Control Point wants to set up QoS for all traffic streams irrespective of the *IpProtocol*.

2.2.2.2.2.6. v2TrafficId

This is an OPTIONAL structure. It MAY contain a *v3TrafficId* structure.

2.2.2.2.2.6.1. v3TrafficId

This is an OPTIONAL structure under the *TrafficId* structure. The *v3TrafficId* structure contains two fields.*SourceUuid* and *DestinationUuid*.

2.2.2.2.2.6.1.1. SourceUuid

This is an OPTIONAL field. The *SourceUuid* field contains the UDN of the UPnP Device at the source.

```
UUID ::= (* The UDN of the device including the prefix 'uuid:'. Refer to
Section 2.1 of [DEVICE] *)
```

Example:

```
uuid:2fac1234-31f8-11b4-a222-08002b34c003
```

2.2.2.2.2.6.1.2. DestinationUuid

This is an OPTIONAL field. The *DestinationUuid* field contains the UDN of the UPnP Device at the destination.

```
UUID ::= (* The UDN of the device including the prefix 'uuid:'. Refer to
Section 2.1 of [DEVICE] *)
```

Example:

```
uuid:1fac1234-31f8-11b4-a222-08002b34c002
```

2.2.2.2.3. AvailableOrderedTspecList

This is a MANDATORY structure. *AvailableOrderedTspecList* contains one or more *Tspec* components. Their value of *TspecIndex* reflects the order of preference. The values of *TspecIndex* MUST be unique within an *AvailableOrderedTspecList*. TSPECs with smaller *TspecIndex* values are more preferred. *TspecIndex* values need not be consecutive numbers. The order of *Tspec* structures in the *AvailableOrderedTspecList* is not relevant.

2.2.2.2.3.1.Tspec

This is a MANDATORY structure. *Tspec* contains a TSPEC (TrafficSpecification) which is a description of the QoS Requirements.

In a UPnP-AV scenario, this information is extracted from the *ContentDirectory* Service of the *MediaServer* Device. In the *ContentDirectory* Service, *Tspec* is represented either as a string, containing an escaped XML structure, or as an URI pointing to the escaped XML structure. The UPnP-AV Control Point uses *CDS:Browse()* and/or *CDS:Search()* action[3] calls to acquire the *Tspec* structure(s) associated with the content.

The XML structure contains the following elements:

- *TspecIndex*: This is a MANDATORY field. It is a unique numerical index associated with a particular *Tspec*. The value of *TspecIndex* indicates preference (as defined by the application or the end user). A *Tspec* with a smaller index is more preferred compared to a *Tspec* with a larger index.
- *AvTransportUri*: This is an OPTIONAL field. It is a string that contains the URI associated with the UPnP-AV content item for which QoS is being requested.
- *AvTransportInstanceId*: This is an OPTIONAL field. It is an integer representing the unique *InstanceID* associated with the UPnP *AVTransport* Service associated with the content item for which QoS is being requested.
- *TrafficClass*: This is an OPTIONAL field. It contains the traffic class associated with the traffic stream. This is an enumerated variable that can be assigned to one of the following list of values:
  - `NetworkControl`
  - `StreamingControl`
  - `Voice`
  - `AV`
  - `Data`
  - `Audio`
  - `Image`
  - `Gaming`
  - `Other`
  - `Background`
- *v2TrafficSpecification*: This is an OPTIONAL structure. It MAY contain a *v3TrafficSpecification* structure. The *v3TrafficSpecification* structure contains the parameters as specified in Section 2.2.2.3.

2.2.2.2.4. ActiveTspecIndex

This is an OPTIONAL field when provided to the *QosManager* Service or *QosPolicyHolder* Service and a MANDATORY field when provided to the *QosDevice* Service. *ActiveTspecIndex* contains an integer which

[3] CDS = *ContentDirectory* Service

indicates the index of the current active *Tspec* from the *AvailableOrderedTspecList*. *Tspec* and *AvailableOrderedTspecList* are defined above.

2.2.2.2.5. *TrafficImportanceNumber*

This is an OPTIONAL field when provided to the *QosManager* Service or *QosPolicyHolder* Service and a MANDATORY field when provided to the *QosDevice* Service for Prioritized and Hybrid QoS requests. A *TrafficImportanceNumber* is an integer with values in the range of 0 through 7. This value follows the numbering scheme for traffic classes as described in IEEE 802.1D Annex G [IEEE 802.1D] and with additional traffic classes described in section 2.4.1.4.2. This value is used by *QosDevice* service(s) in the traffic's path to indicate what priority level to utilize when priority tagging the traffic's network packets.

2.2.2.2.6. *QosBoundarySourceAddress*

This is an OPTIONAL field. If a traffic stream originates outside the UPnP-QoS network's subnet (e.g. on the Internet), the *QosBoundarySourceAddress* will be treated as the QoS termination point for UPnP-QoS. This parameter is OPTIONAL, because it is applicable only to traffic streams originating outside the home network. It should be noted that this address is not part of the traffic identifier, because the IP packets will carry the IP address of the actual source address. *QosBoundarySourceAddress* is used by the *QoS Manager* for decisions related to path determination and device selection.

The "Don't care" situation that may apply for *SourceAddress* and *DestinationAddress* IP address does not apply here.

2.2.2.2.7. *QosBoundaryDestinationAddress*

This is an OPTIONAL field. If a traffic stream terminates outside the UPnP-QoS network's subnet (e.g. on the Internet), the *QosBoundaryDestinationAddress* will be treated as the QoS termination point for UPnP-QoS. This parameter is OPTIONAL, because it is applicable only to traffic streams terminating outside the home network. It should be noted that this address is not part of the traffic identifier, because the IP packets will carry the IP address of the actual destination address. *QosBoundaryDestinationAddress* is used by *QoS Manager* for decisions related to path determination and device selection.

The "Don't care" situation that may apply for *SourceAddress* and *DestinationAddress* IP address does not apply here.

2.2.2.2.8. *MediaServerConnectionId*

This is an OPTIONAL field. This field may be useful in case of UPnP-AV based streaming with multiple *Tspec* structures. The *MediaServerConnectionId* may be used by the *QosManager* Service to identify the traffic stream that is being setup by the UPnP-AV Control Point which is requesting QoS. Please refer to UPnP-QoS Architecture document for more details [QoS Architecture]. The *MediaServerConnectionId* is obtained from the *MediaServer* Device via the *CM:PrepareForConnection()* action[4].

2.2.2.2.9. *MediaRendererConnectionId*

This is an OPTIONAL field. This field may be useful in case of UPnP-AV based streaming with multiple *Tspec* structures. *MediaRendererConnectionId* may be used by the *QosManager* Service to identify the traffic stream that is being setup by the UPnP-AV Control Point which is requesting QoS. Please refer to UPnP-QoS Architecture document for more details [QoS Architecture]. The *MediaRendererConnectionId* is obtained from the *MediaRenderer* Device via the *CM:PrepareForConnection()* action.

---

[4] CM = UPnP-AV *ConnectionManager* Service

2.2.2.2.10.TrafficLeaseTime

This is an OPTIONAL field. This field contains the lease time associated with a particular traffic stream. The lease time is expressed in seconds. When not specified (which MUST only occur for prioritized streams), it indicates an indefinite lease, that is the *TrafficDescriptor* will remain active until it is explicitly removed or the *QosDevice* is rebooted. When the *QosDevice* Service or *QosManager* Service reports the lease time to a Control Point, this is the remaining lease time.

2.2.2.2.11.v2

This is an OPTIONAL structure. It contains elements that are defined in UPnP-QoS v2 as specified below.

2.2.2.2.11.1.PolicyHolderId

This is an OPTIONAL field. A *PolicyHolderId* is a value that uniquely identifies *QosPolicyHolder* Service on the home network. This value informs the *QosManager* Service which *QosPolicyHolder* Service to query for a *TrafficPolicy*. The value must be of the following form as described using EBNF:

```
PolicyHolderId ::= UUID ':' serviceId

UUID ::= (* The UDN of the QosPolicyHolder service including the prefix
'uuid:'. Refer to Section 2.1 of [DEVICE] *)

serviceId ::= (* The serviceId of the QosPolicyHolder service including the
prefix 'urn:upnp-org:serviceId:'. Refer to Section 2.1 of [DEVICE] *)
```

For example:

```
uuid:2fac1234-31f8-11b4-a222-08002b34c003:urn:upnp-
org:serviceId:QosPolicyHolder-3a
```

2.2.2.2.11.2.PolicyLastModified

This is an OPTIONAL field. *PolicyLastModified* is an RFC3339 compliant string [RFC 3339] that identifies the wall clock date/time when the policy on a device containing the *QosPolicyHolder* Service was last modified. It is recognized that device clocks may not be synchronized. This field is valuable for implementations to provide diagnostic information to the end user.

2.2.2.2.11.3.PolicyModifyingUserName

This is an OPTIONAL field. *PolicyModifyingUserName* is a string that identifies the user who last changed the policy. This field MUST be <= 64 UTF-8 characters. At present this specification does not define the semantics for this field. Examples include, “Dad”, “Jimmy”, etc. This field is valuable for implementations to provide diagnostic information to the end user.

2.2.2.2.11.4.PolicyHolderConfigUrl

This is an OPTIONAL field. *PolicyHolderConfigUrl* is a string containing a URL on the device that provides the *QosPolicyHolder* Service. If the device provides the URL, containing "HTTP" or "HTTPS" as the <protocol>, for its configuration, then the control point may retrieve a page using the provided URL to allow a user to configure the *QosPolicyHolder* Service (the configuration method is out-of-scope). The host portion of the URL must be in dotted-decimal IP address form, e.g. "http://10.0.0.131/ config_policy.html".

```
PolicyHolderConfigURL ::= Protocol '://' Host [':' Port] [Path ['?' Query]]
```

2.2.2.2.11.5.v3

This is an OPTIONAL structure. It contains elements that are defined in UPnP-QoS v3 as specified below.

2.2.2.2.11.5.1. QosBoundarySourceUuid

This is an OPTIONAL field. It contains the UDN of the UPnP Device at the QoS Boundary Source.

2.2.2.2.11.5.2. QosBoundaryDestinationUuid

This is an OPTIONAL field. It contains the UDN of the UPnP Device at the QoS Boundary Destination.

2.2.2.2.11.5.3. Criticial

This is an OPTIONAL field. The *Critical* field contains a Boolean that expresses the control point's belief on whether this traffic stream is critical. Streams for entertainment are not critical traffic streams. Examples of critical traffic streams are emergency alerts, emergency calls, etc.

2.2.2.2.11.5.4. PolicyHolderConsultedId

This is an OPTIONAL field. The *PolicyHolderConsultedId* is a string that contains a value that uniquely identifies the *QosPolicyHolder* Service that was consulted in setting up QoS for the stream. The format is the same as the format of the *PolicyHolderId* field.

2.2.2.2.11.5.5. PolicyHolderConsultedType

This is an OPTIONAL field. The value of this integer expresses which version of the behavior to determine which *QosPolicyHolder* Service to be used was used for this traffic stream. There is no default value.

Table 2-2: Meaning of *PolicyHolderConsultedType* parameter

Value	Meaning
0	Default policy
1	There was exactly one *QosPolicyHolder* Service on the network and that one was used. This is the original UPnP-QoS v1 behavior.
2	The *QosPolicyHolder* Service that was used is the one indicated by the Control Point. This is the normal UPnP-QoS v2 behavior.
3	The *QosPolicyHolder* Service that was used is the one preferred by the user.

2.2.2.2.11.5.6. UpdateTrafficLeaseTime

This is an optional **boolean** field. When a Control Point sets this to "1" in a QoS update request, it indicates that the Control Point is requesting that only the *TrafficLeaseTime* is updated for the traffic stream.

2.2.2.2.12. OptionalPolicyParameters

This is an OPTIONAL structure. It consists of the fields described in subsequent sections below.

2.2.2.2.12.1. UserName

This is an OPTIONAL field. *UserName* contains a string associated with the user who requested a particular traffic stream. This field identifies the name of the user initiating the UPnP-QoS Action. The field MUST be <= 64 UTF-8 characters. At present this specification does not define the semantics for this field. This field represents the name of the user initiating the QoS action. Examples include, "Dad", "Jimmy", "CN#1234567". The latter is an example of a name provided by a service provider to uniquely identify its individual customers.

2.2.2.2.12.2.CpName

This is an OPTIONAL field. *CpName* contains a string associated with the Control Point requesting QoS for the traffic stream. This field identifies the name of the Control Point initiating the UPnP-QoS Action. The field MUST be <= 64 UTF-8 characters. At present this specification does not define the semantics for this field. The *CpName* may specify the brand name of the Control Point or it may indicate the location of the Control Point such as "living room".

2.2.2.2.12.3.VendorApplicationName

This is an OPTIONAL field. *VendorApplicationName* contains a single URI string associated with an application. This field identifies the name of the application initiating the UPnP-QoS Action. For applications specified by vendors, the value of this field MUST begin with "urn:", followed by an ICANN domain name owned by the vendor, followed by ":application:", followed by an application name, i.e. the value of this field MUST be constructed as follows:

```
VendorApplicationName ::= 'urn:' DomainName*  ':' Application

DomainName ::= (* Application vendor domain name. This MUST follow the syntax
specified for Namespace Identifier (NID) in [RFC 2141]. The domain-name field
MUST be <= 64 UTF-8 characters.  If the control point is not aware of this
field, then it MAY be left blank *)

Application ::= 'application:' AppName

AppName ::= (* Name of the application provided by a vendor. This must follow
the syntax specified for Name Specific String (NSS) in [RFC 2141]. The
AppName field MUST be <= 64 UTF-8 characters. The AppName MUST NOT contain a
colon character. *)
```

For example:

```
urn:example-org:application:MyApp
urn:example-org:application:ScheduledRecording
```

When requesting QoS it may be useful to know the manufacturer and name of the application software client or server that will process the traffic stream.

2.2.2.2.12.4.PortName

This is an OPTIONAL field. *PortName* contains a single URI **string** associated with the fixed port used by an application for its protocol set-up. This field, if present, shall identify the port number used by the application. The value of this field MUST begin with "urn:", followed by an ICANN domain name owned by the vendor, followed by ":port:", followed by the port name. i.e. the value of this field MUST be constructed as follows:

```
PortName ::= 'urn:' DomainName*  ':' Port

DomainName ::= (* Application vendor domain name. This MUST follow the syntax
specified for Namespace Identifier (NID) in [RFC 2141]. The domain-name field
MUST be <= 64 UTF-8 characters.  If the control point is not aware of this
field, then it MAY be left blank *)

Port ::= 'port:' PortNumber

PortName ::= (* An integer value.  *)
```

For example:

```
urn:example-org:port:80
```

PortNumber is a fixed port number used by the application for protocol set-up. If the application uses a range of port numbers then the starting port number of that range MUST be specified. This value is either a vendor specific port used for the application or an IANA assigned port number for the application. The list of all the IANA assigned port numbers is maintained by IANA (see [IANA Portnumbers]).

This field indicate the port that is used by an application for its connection set-up. The use of IANA assigned port numbers is RECOMMENDED, but if not available for a particular application, a vendor assigned port number MAY be used.

2.2.2.2.12.5. ServiceProviderServiceName

This is an OPTIONAL field. *ServiceProviderServiceName* field contains a single URI string associated with a service offered by a service provider. This field identifies the name of the service offered by a service provider. For services provided by service providers, the value of this field MUST begin with "urn:", followed by an ICANN domain name owned by a service provider, followed by ":service:", followed by a service name, i.e. the value of this field MUST be constructed as follows:

```
ServiceProviderServiceName ::= 'urn:' DomainName*  ':' Service

DomainName ::= (* Application vendor domain name. This MUST follow the syntax
specified for Namespace Identifier (NID) in [RFC 2141]. The domain-name field
MUST be <= 64 UTF-8 characters.  If the control point is not aware of this
field, then it MAY be left blank *)

Service ::= 'service:' ServName

ServName ::= (* Name of the service provided by a service vendor. This must
follow the syntax specified for Name Specific String (NSS) in [RFC 2141]. The
ServName field MUST be <= 64 UTF-8 characters. The ServName MUST NOT contain
a colon character. *)
```

For example:

```
urn:example-org:service:VoD
```

2.2.2.3. Description of fields in v3TrafficSpecification structure

This section defines the parameters of the *v3TrafficSpecification* structure which is used to characterize the requirements of a specific traffic stream. The traffic specification describes the stream as it flows from Layer 3 to Layer 2. Therefore it includes headers such as the IP header, TCP header, or UDP header but excludes the MAC header, the LLC header, the IEEE 802.1q header, lower-layer FEC bits, or physical layer overhead.

The fields of the *v3TrafficSpecification* structure are listed in Table 2-3: Traffic Specification Parameters together with a reference to their normative definition.

Table 2-3: Traffic Specification Parameters

Name	M/O[5]	Definition	Default
RequestedQosType	O	Section 2.2.2.3.1	0 (zero)
DataRate	M (if *RequestedQosType* non-zero)	[RFC 2212] where it is called bucket rate *r*.	N/A

[5] M = MANDATORY, O = OPTIONAL, X=Non-Standard.

Name	M/O[5]	Definition	Default
MinServiceRate	O	Section 2.2.2.3.7	None
MaxBurstSize	O	[RFC 2212] where it is called bucket depth *b*.	None
PeakDataRate	O	[RFC 2212] where it is called peak rate *p*.	Same as *DataRate*.
ReservedServiceRate	O	[RFC 2212] where it is called Rspec rate term *R*.	None
TimeUnit	O	Section 2.2.2.3.4	1000 µsec
MaxPacketSize	O	[RFC 2212] where it is called maximum datagram size *M*.	1500 octets
E2EMaxDelayHigh	O	Section 2.2.2.3.11	None
E2EMaxDelayLow	O	Section 2.2.2.3.13	None
E2EMaxJitter	O	Section 2.2.2.3.12	None
QosSegmentMaxDelayHigh	O	Section 2.2.2.3.14	None
QosSegmentMaxDelayLow	O	Section 2.2.2.3.14.5	None
QosSegmentMaxJitter	O	Section 2.2.2.3.14.4	None
MaxServiceInterval	O	Section 2.2.2.3.15	None
MinServiceInterval	O	Section 2.2.2.3.16	None
LossSensitivity	O	[RFC 1363]	0
ServiceType	O	Section 2.2.2.3.18	1

We will introduce the parameters with examples after which the definition follows.

2.2.2.3.1. RequestedQosType

The optional *RequestedQosType* parameter, if supplied, MUST contain a non-negative integer value. It is used to indicate the type of Quality of Service that is requested with this TrafficDescriptor. The default value is 0.

Table 2-4: Meaning of *RequestedQosType* parameter

Value	Meaning
0	Prioritized QoS If the QoS (update) request succeeds, this traffic stream MUST be successfully set up using prioritized QoS for every QoS segment it traverses. This includes, in particular, prioritized QoS setup for QoS segments that support parameterized QoS.

Value	Meaning
1	Hybrid QoS If the QoS (update) request succeeds, this traffic stream MUST be successfully admitted using parameterized QoS on every parameterized QoS segment it traverses and MUST be set up using prioritized QoS on every prioritized QoS segments it traverses.
2	Parameterized QoS If the QoS (update) request succeeds, this traffic stream MUST be successfully admitted using parameterized QoS on every parameterized QoS segments it traverses and the traffic stream MUST NOT traverse any non-parameterized QoS segment.
3 and higher	Reserved

2.2.2.3.2. Data rate related parameters

The following sections introduce the parameters related to data rate. These sections are for information only and an attempt to explain the RFCs that contain the normative definitions.

For example, consider a traffic stream that emits *x(t)* octets at time instant *t*. For simplicity we ignore propagation delay and therefore when a single packet is available at *t=1*, we have *x(1)* equal the packet size. The time ranges over the real numbers but the traffic stream is sent in packets and therefore *x(t)=0* almost everywhere. For ease of notation a stream starts at *t=0*.

2.2.2.3.2.1. Example

A simple example of the traffic generated by a traffic stream is shown in Figure 1. In this case, the amount of data and the frequency it is sent with is constant. Formally, there is a $C > 0$ and a time interval $s > 0$, such that for every positive integer k we have $x(ks) = C$ and for every $t > 0$ that is not an integer multiple of s, we have $x(t) = 0$.

This traffic stream will be modeled by setting the *DataRate* parameter, that we will describe below, to *C*.

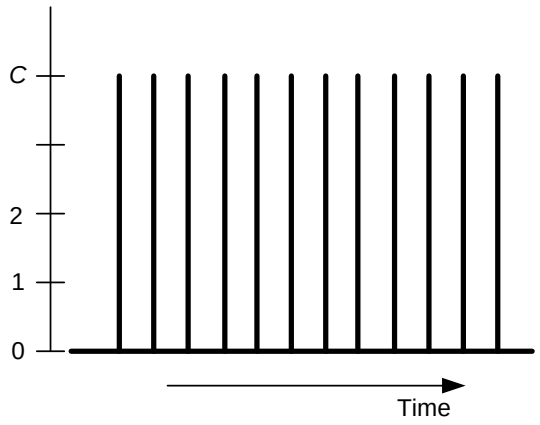

Figure 1 An example traffic stream

If this traffic stream is served with a rate below C, packets will accumulate in queues. The queues can be emptied after the end of the traffic stream, when no new packets arrive.

If this traffic stream is served at a rate of at least *C* then packets are not delayed in queues and the total delay is the propagation delay. In Section 2.2.2.3.11 we describe delay in more detail.

2.2.2.3.3. Data Rate

2.2.2.3.3.1. Definition and Requirements (Normative)

The parameter *DataRate* is defined in [RFC 2212] where it is called bucket rate *r*. The parameter *DataRate* MUST be provided if the value of the *RequestedQosType* parameter is non-zero. Its unit is octets per second.

2.2.2.3.3.2. Explanation of the Definition (Informative)

The parameter *DataRate* intends to characterize the long term data rate. To ones skilled in the art it is also called Mean Data Rate or Bucket Rate. For a traffic stream of a finite duration *T*, the *DataRate* is defined as

$$\frac{1}{T}\sum_{0<=t<=T} x(t)$$

The unit of *DataRate* is therefore octets per second.

A crude approximation for a stored file, is obtained by dividing the file length by the duration of the play out of that file.

For traffic streams of infinite duration such as those output from a tuner, the *DataRate* is defined as

$$\lim_{T\to\infty}\frac{1}{T}\sum_{0<=t<=T} x(t)$$

For practical purposes a sufficiently large value of *T* is used to determine the *DataRate*.

Since serving a traffic stream with a rate below the *DataRate* will lead to an unbounded accumulation of packets in queues, the traffic stream MUST be served with a rate of at least *DataRate*.

2.2.2.3.3.3. Example Constant Bit Rate Traffic

In this section we present an example of using the *DataRate* for a traffic stream with a traffic corresponding to Figure 2. As can be seen from the figure, the amount of data and the frequency it is sent with is constant. Formally, there is a $C > 0$ and a time interval $s > 0$, such that for every positive integer k we have $x(ks) = C$ and for every $t > 0$ that is not an integer multiple of s, we have $x(t) = 0$.

In this example, the *DataRate* is *C*.

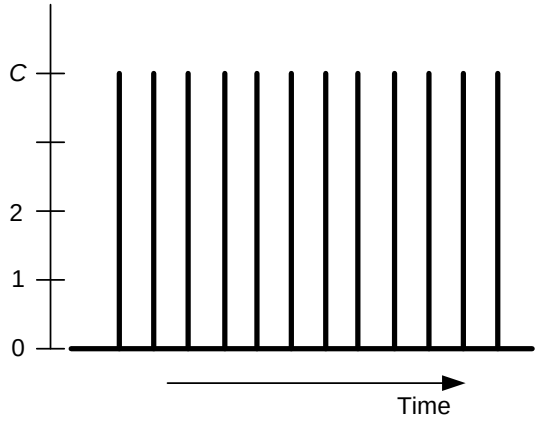

Figure 2 A traffic stream with a constant bit rate

2.2.2.3.3.4. Example Variable Bit Rate Traffic

In this section we present an example of using the *DataRate* for a traffic stream with a traffic corresponding to Figure 3 (left side). In contrast to the previous example, the amount of data is not constant. We can calculate *DataRate* as 3.08 or precisely

$$\frac{1}{12}(4+2+4+1+3+4+4+2+4+1+4+4).$$

If this traffic stream is served at a rate equal to *DataRate*, then queues will fill if $x(t) >$ *DataRate* and will empty if $x(t) <$ *DataRate*. An efficient way of dealing with this traffic stream is by specifying additional parameters as defined in, e.g., Section 2.2.2.3.5. However, it may be the case that not all details of the traffic are known.

If we do not know more about the stream, and if we are not worrying too much about efficiency, we can approximate our traffic stream with another traffic stream that has a constant bit rate D, see Figure 3 (right side). That is, formally there is a $D > 0$ and a time interval $s > 0$, such that for every positive integer k we have $x(ks) \leq D$ and for every $t > 0$ that is not an integer multiple of s, we have $x(t) = 0$.

In this case, we pick the larger value of D and specify the traffic stream as if it is a constant bit rate stream of *DataRate* D.

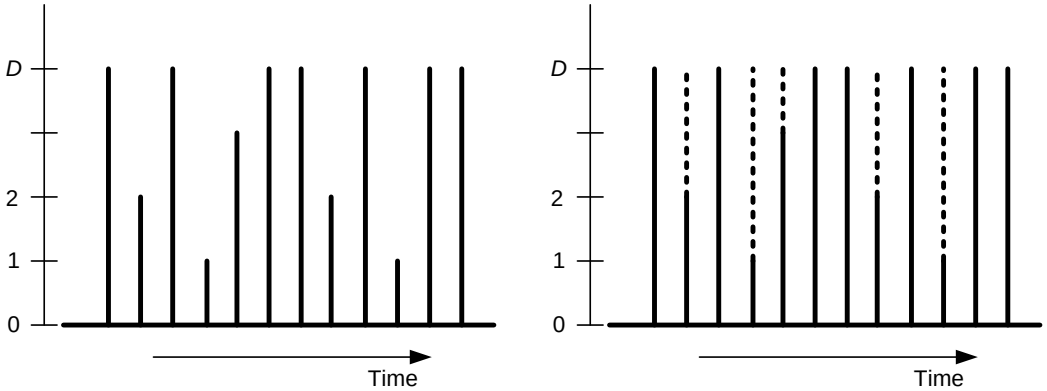

Figure 3 The non-constant bit-rate traffic (left) can be viewed as a constant bit-rate traffic of a higher constant bit-rate *D* (right)

In certain contexts, this traffic is characterized by a Peak Data Rate. In Section 2.2.2.3.5 below we define *PeakDataRate* and show how the combination of *DataRate* and *PeakDataRate* can be used to schedule this type of traffic in a more efficient way.

2.2.2.3.4. Time Unit (Normative)

This integer field specifies the smallest time interval, in units of microseconds, in which the behavior of the traffic source is characterized in this Traffic Specification. The value of the parameter *TimeUnit* will be used in the specification of the *PeakDataRate* and *MaximumBurstSize* below. Stated differently the *TimeUnit* parameter is a sampling interval.

2.2.2.3.4.1. More information on Time Unit (Informative)

An example *TimeUnit* is 1024 µs which is the one specified in IEEE 802.11. However, in many cases it is more convenient to choose a *TimeUnit* that is traffic specific rather than Layer-2 specific. A translation between Traffic Specifications from one *TimeUnit* to another can be performed by a *QosManager* Service and an example algorithm can be found in the [QoS Architecture]. For instance, the sampling rate or frame rate for audio/voice applications, and the 90khz reference clock rate for broadcast quality A/V applications, are good choices for the *TimeUnit* as it makes it easy to derive the Traffic Specification parameters.

The smaller the *TimeUnit* finer the granularity in which the token bucket Traffic Specification characterizes the traffic source. However, Traffic Specification parameters with finer granularity are more difficult to obtain.

An example of different Traffic specifications with different values for the *TimeUnit* parameter follows the definition of *PeakDataRate*.

2.2.2.3.5. Peak Data Rate

2.2.2.3.5.1. Definition and Requirements (Normative)

The parameter *PeakDataRate* is defined in [RFC 2212] where it is called peak rate *p*. The parameter *PeakDataRate* is OPTIONAL. If not provided it defaults to the value of *DataRate*. The *PeakDataRate* SHOULD be specified to accommodate packet quantization.

2.2.2.3.5.2.Explanation of the Definition (Informative)

The parameter *PeakDataRate* contains the peak date rate and intends to characterize the largest data rate of the traffic stream. The physics of the network teaches us that every packet will be send at the wire rate. It is therefore crucial to define *PeakDataRate* over a longer interval of time than the duration of sending a single packet. This interval should not be too long either, because the longer the interval, the closer *PeakDataRate* gets to *MeanDataRate*. The interval is *TimeUnit*.

Let *p* be the smallest value that for every interval $[t_1, t_2]$ of length at least *TimeUnit* satifies

$$\frac{1}{t_2 - t_1} \sum_{t_1 \le t \le t_2} x(t) \le p$$

Then *p* is called the *PeakDataRate*. The unit of *PeakDataRate* is octets per second, even though it is calculated per *TimeUnit*. If the *PeakDataRate* cannot be reliably calculated, a larger value of *p* will also do as it will not endanger the guaranteed level of QoS. It will, however, have a negative impact on the efficiency of the traffic stream's transportation.

It should be noted that *p* is not the same as the maximum of *x(t)*.

The principal purpose of *PeakDataRate* is that it defines an upper bound on the data rate at which this traffic stream needs to be served in order to serve the traffic stream within time. If the traffic stream is served at a rate above *PeakDataRate*, resources are unnecessarily reserved and possibly wasted.

2.2.2.3.6. Maximum Burst Size

2.2.2.3.6.1. Definition and Requirements (Normative)

The parameter *MaxBurstSize* is defined in [RFC 2212] where it is called bucket depth *b*. The parameter *MaxBurstSize* is OPTIONAL. If not provided it defaults to the value 0 (zero). The *MaxBurstSize* should be specified to accommodate packet quantization.

2.2.2.3.6.2. Explanation of the Definition (Informative)

The parameter *MaxBurstSize* contains the maximum burst size.

Let *b* be the smallest value that for every interval $[t_1, t_2]$ of length at least *TimeUnit* satisfies.

$\sum_{t_1 \le t \le t_2} x(t) \le r(t_2 - t_1) + b$, where *r* is Data Rate

Then *b* is called the *MaxBurstSize*. The unit of *MaxBurstSize* is octets. If $b = 0$, then the traffic stream is a constant bit rate stream and has no bursts.

In graphical terms, *b* is the surface between the graph of x and the horizontal line $y = r$ (restricted to the interval).

The principal purpose of *MaxBurstSize* is that it defines an upper bound on the buffer size if the traffic stream is served at a rate between mean and peak data rate. If the buffer is at least *b*, packets need not be dropped.

2.2.2.3.7. MinServiceRate

The parameter *MinServiceRate* is OPTIONAL. There is no default value. It specifies the lowest data rate, in octets per second, required for transport of the packets belonging to the TrafficStream. See Minimum data rate in [HPAV] for an example usage.

2.2.2.3.8. ReservedServiceRate

2.2.2.3.8.1. Definition and Requirements (Normative)

The parameter *ReservedServiceRate* is defined in [RFC 2212] where it is called Rspec rate term *R*. The parameter *ReservedServiceRate* is OPTIONAL. There is no default.

2.2.2.3.8.2. Explanation of the Definition (Informative)

This data rate typically lies between *DataRate* and *PeakDataRate* and provides a good indication to the L2 Technology. The larger *ReservedServiceRate* the smaller the delay will be. An example usage of *ReservedServiceRate* can be found in Section 9.2 of the [PACKETC].

2.2.2.3.9. Packet Size Parameters

Packet-size parameters such as *MaxPacketSize* convey the information about packetization of traffic streams by an application. All the packet-size parameters include the application data and an approximation of the overhead

of all protocol headers at and above the IP layer (IP, TCP, UDP, RTP, etc.).This approximation is of the overhead of all protocol headers is performed by the application. They MUST not include the L2 header size, because these headers may change in size as the packet crosses different QoS Segments.

2.2.2.3.10.MaxPacketSize

2.2.2.3.10.1.Definition and Requirements (Normative)

The parameter *MaxPacketSize* is defined in [RFC 2212] where it is called maximum datagram size *M*. The parameter *MaxPacketSize* is OPTIONAL. If not provided it defaults to the value 1500 octets.

2.2.2.3.10.2.Explanation of the Definition (Informative)

This parameter specifies the maximum size, in units of octets, of the IP packets belonging to the traffic stream in this traffic specification. It is typically used to derive a queuing delay bound such as the Parekh-Gallager delay bound.

2.2.2.3.11.E2EMaxDelayHigh

2.2.2.3.11.1.Definition and Requirements (Normative)

The *E2EMaxDelayHigh* parameter defines the desired upper bound of the End-to-End Delay (as defined in section 1.1.2.2.12) of individual packets belonging to the traffic stream. The parameter *E2EMaxDelayHigh* is OPTIONAL. The unit in which the *E2EMaxDelayHigh* parameter is expressed is microseconds.

2.2.2.3.11.2.Explanation of the Definition (Informative)

The Control Point supplies this parameter to request a specific upper bound to End-To-End Delay.

If this parameter is supplied by the Control Point, then the *QoS Manager* ensures that the sum of the maximum QoS Segment Delay values committed by various *QosDevice* services on the path is less than *E2EMaxDelayHigh* for a successful admission or update request.

If this parameter is not provided by the Control Point then it indicates that any End-to-End delay is acceptable and the *QosManager* Service does not include QosSegmentMaxDelayHigh and QosSegmentMaxDelayLow parameters in the QoS setup/update request to a *QosDevice* service.

2.2.2.3.12.E2EMaxJitter

2.2.2.3.12.1.Definition and Requirements (Normative)

The *E2EMaxJitter* parameter defines an UPPER bound to the End-to-End Jitter as defined in section 1.1.2.2.13. The *E2EMaxJitter* parameter is OPTIONAL. The unit in which the *E2EMaxJitter* parameter is expressed is microseconds.

2.2.2.3.12.2.Explanation of the Definition (Informative)

The Control Point supplies this parameter to request bounded End-to-End Jitter .

If this parameter is supplied by the Control Point, then the *QoS Manager* ensures that the sum of the maximum QoS Segment Jitter values committed by various *QosDevice* services on the path is less than *E2EMaxJitter* for a successful admission or update request.

If E2EMaxJitter parameter is not provided, any finite End-to-End Jitter, up to and including the *E2EMaxDelayHigh*, is acceptable and the *QosManager* Service does not include QosSegmentMaxJitter parameter in the QoS setup/update request to a *QosDevice* service.

2.2.2.3.13.E2EMaxDelayLow

2.2.2.3.13.1.Definition and Requirements (Normative)

The *E2EMaxDelayLow* parameter is a LOWER bound for the maximum End-to-End Delay. The parameter *E2EMaxDelayLow* is OPTIONAL. The purpose of the the *E2EMaxDelayLow* parameter is to express that packet delays smaller than *E2EMaxDelayLow* are not necessary.

The unit in which the *E2EMaxDelayLow* parameter is expressed is microseconds.

2.2.2.3.13.2.Explanation of the Definition (Informative)

This parameter is used together with the *E2EMaxDelayHigh* parameter to provide guidance to the *QosManager* Service on the desired delay. The *E2EMaxDelayLow* parameter is intended to avoid unnecessarily low delay requirements. For more information, see the definition of *QosSegmentMaxDelayLow* parameter in section 2.2.2.3.14.5

Note: *E2EMaxDelayLow* is NOT the same concept as minimum of End-to-End delay. In other words *E2EMaxDelayLow* does NOT state that every packet has an End-to-End Delay of at least *E2EMaxDelayLow*.

2.2.2.3.14.QosSegmentSpecificParameters

This is an OPTIONAL structure. It contains TSPEC parameters that are specific to a QoS Segment.

2.2.2.3.14.1.InterfaceId

This is a MANDATORY field. The value is of type string and MUST uniquely identify an interface within a *QosDevice* Service.

2.2.2.3.14.2.QosSegmentId

This is a MANDATORY field that contains the *QosSegmentId* of the QoS Segment for which the *QosSegmentSpecificParameters* are provided.

2.2.2.3.14.3.QosSegmentMaxDelayHigh

This parameter is related to QoS Segment specific parameters.

2.2.2.3.14.3.1. Definition and Requirements (Normative)

The *QosSegmentMaxDelayHigh* parameter defines an upper bound of the maximum of QoS Segment Delay (as defined in (section 1.1.2.2.38) of individual packets belonging to the traffic stream. The parameter *QosSegmentMaxDelayHigh* is OPTIONAL. The unit in which the *QosSegmentMaxDelayHigh* parameter is expressed is microseconds.

2.2.2.3.14.3.2. Explanation of the Definition (Informative)

This parameter is optionally supplied as one of the input TSPEC parameters to a *QosDevice* service for admission or update of a traffic stream. If the *QoS Manager* does not provide this parameter in the TSPEC, any

delay is acceptable. In this case, the *QosDevice* Service may use the best upper bound for the delay within the QoS segment.

If the control point has specified an *E2EMaxDelayHigh* value the *QoS Manager* determines *QosSegmentMaxDelayHigh* values to use as input for admission or update actions on the *QosDevice* service. A simple example is to set *QosSegmentMaxDelayHigh* = *E2EMaxDelayHigh* / (Number of QoS Segments on the Path). Alternatively, the *QD:GetExtendedQosState()* action optionally provides *QosSegmentMaxDelayHigh* parameter to the *QoS Manager* as one the ProtoTspec parameters [QOS DEVICE] . The *QosManager* Service examines if the sum of the *QosSegmentMaxDelayHigh* values returned in the ProtoTspec for all QoS segments is smaller than the *E2EMaxDelayHigh* value to determine if the request for admission or update would be successful.

2.2.2.3.14.4.QosSegmentMaxJitter

This parameter is related QoS Segment specific parameters.

2.2.2.3.14.4.1. Definition and Requirements (Normative)

The QoS Segment Jitter is defined as the absolute value of the difference between the MAXIMUM QoS Segment Delay and the MINIMUM QoS Segment Delay. The *QosSegmentMaxJitter* parameter is the UPPER bound to the QoS Segment Jitter.

The *QosSegmentMaxJitter* parameter is OPTIONAL. The unit in which the *QosSegmentMaxJitter* parameter is expressed is microseconds.

2.2.2.3.14.4.2. Explanation of the Definition (Informative)

This parameter is optionally supplied as one of the input TSPEC parameters to a *QosDevice* service for admission or update of a traffic stream. If the *QosSegmentMaxJitter* parameter is specified, then QoS Segment Jitter will be bounded and this can be used to determine buffer space at the egress device, and it can also be used to bound the *E2EMaxJitter*.

If the control point has specified an *E2EMaxJitter* value, the *QoS Manager* MAY (but is NOT REQUIRED to) determine *QosSegmentMaxJitter* values to use as input for admission or update. A simple example is to set *QosSegmentMaxJitter* = *E2EMaxJitter* / (Number of QoS Segments on the Path). Alternatively, the *QD:GetExtendedQosState()* action optionally provides the *QosSegmentMaxJitter* parameter to the *QoS Manager* as one the ProtoTspec parameters [QOS DEVICE] . The *QosManager* Service examines if the sum of the *QosSegmentMaxJitter* values returned in the ProtoTspec for all QoS segments is smaller than the *E2EMaxJitter* value to determine if the request for admission or update would be successful.

If the *QosSegmentMaxJitter* parameter is not specified, the *QosDevice* Service MAY infer jitter requirements (for example, from the *TrafficClass* value).

2.2.2.3.14.5.QosSegmentMaxDelayLow

This parameter is related to QoS Segment specific parameters.

2.2.2.3.14.5.1. Definition and Requirements (Normative)

The *QosSegmentMaxDelayLow* parameter is an informative LOWER bound for the MAXIMUM of QoS Segment Delay. The main purpose of the the *QosSegmentMaxDelayLow* parameter is to express that packet delays smaller than *QosSegmentMaxDelayLow* are not necessary. The parameter *QosSegmentMaxDelayLow* is OPTIONAL. The unit in which the *QosSegmentMaxDelayLow* parameter is expressed is microseconds.

2.2.2.3.14.5.2. Explanation of the Definition (Informative)

This parameter is used together with the *QosSegmentMaxDelayHigh* parameter to provide guidance to the *QosDevice* Service of the expected delay. The *QosSegmentMaxDelayLow* parameter is intended to avoid unnecessarily tight minimum delay requirements on the underlying QoS segments.

Note: *QosSegmentMaxDelayLow* is NOT the same concept as minimum of QoS Segment Delay. In other words *QosSegmentMaxDelayLow* does NOT state that every packet has a QoS Segment Delay of at least *QosSegmentMaxDelayLow*.

2.2.2.3.14.5.3. Example

A certain L2 Technology offers two main classes for parameterized QoS. One class offers delay guarantees in the order of microseconds (class A) and the other class offers delay guarantees in the order of milliseconds (class B). Class A is intended for gaming applications, interactive applications, etc. If the *QosSegmentMaxDelayHigh* parameter is not specified the *QosDevice* Service could always pick class A but this is wasteful. Alternatively, the *QosDevice* Service could determine the appropriate class A or B from for instance the Traffic Class, but this may not work out End-to-End. With the *QosSegmentMaxDelayLow* specified, the *QoS Manager* can take End-to-End conditions into account and help the *QosDevice* Service in determining the appropriate class. The *QD:GetExtendedQosState()* action informs the *QoS Manager* on the options.

2.2.2.3.15. MaxServiceInterval

The parameter *MaxServiceInterval* is OPTIONAL. There is no default value. It specifies the maximum interval, in units of microseconds, between the start of two successive contiguous transmissions of one or more IP packets belonging to the TrafficStream. See Maximum Inter-TXOP time in [HPAV] for an example usage.

2.2.2.3.16. MinServiceInterval

The parameter *MinServiceInterval* is OPTIONAL. There is no default value. It specifies the minimum interval, in units of microseconds, between the start of two successive contiguous transmissions of one or more IP packets belonging to the TrafficStream. See Minimum Inter-TXOP time in [HPAV] for an example usage.

2.2.2.3.17. LossSensitivity

LossSensitivity is an OPTIONAL parameter that specifies whether the stream is sensitive to packet loss . If not provided it defaults to the value 0 (zero). The value 0 (zero) indicates that the traffic stream is insensitive to loss. The value 1 (one) indicates that the traffic stream is sensitive to loss. If the value is a number larger than one, it equals the number of losses of packets of size *MaxPacketSize* for the traffic stream in an interval of 1 second. For more information, see [RFC 1363].

2.2.2.3.18. ServiceType

2.2.2.3.18.1. Definition and Requirements (Normative)

The *ServiceType* parameter is an OPTIONAL Boolean. The default is “1”. If the value is “1”, “guaranteed Quality of Service” as defined in [RFC 2212] is requested. If the value is “0”, “controlled-load service” as defined in [RFC 2211] is requested.

2.2.2.3.18.2. Explanation of the Definition (Informative)

The use of controlled-load service provides flexibility in the specification of QoS requirements in terms of service quality. When the controlled-load service type is specified, the request is to provide the service quality that approximates the network behavior achieved from “Best Effort” service under unloaded conditions.

The result of specifying the controlled-load service type is that network devices and elements in the path can be expected to deliver a very high percentage of packets (packet loss approximates basic packet error rates for the transmission medium) and have a transit delay that for a very high percentage of transmitted packets will not greatly exceed the minimum transit delay experienced by any successfully delivered packet at the speed of light.

If ServiceType is "0", i.e., when controlled-load service is requested *ReservedServiceRate* and the delay parameters are not applicable and MUST not be specified.

2.2.2.4. Relationship between the fields in the **TrafficDescriptor** *structure*

If *SourceAddress* is not provided, the *SourceUuid* MUST be valid.

If *DestinationAddress* is not provided, the *DestinationUuid* MUST be valid.

If the *TrafficDescriptor* contains at least one TSPEC for which *RequestedQosType* > 0 , then

- the *SourceAddress* MUST NOT equal 0.0.0.0, and
- the *DestinationAddress* MUST NOT equal 0.0.0.0, and
- the *TrafficLeaseTime* MUST be specified
- the *DataRate* parameter MUST be specified

2.2.2.5. Sample argument XML string

The following is an example TrafficDescriptor that a Control Point could create to request parameterized QoS at the *QosManager* Service.

```
<?xml version="1.0" encoding="UTF-8"?>
<TrafficDescriptor xmlns="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd
http://www.upnp.org/schemas/qos/ TrafficDescriptor-v3.xsd">
   <TrafficHandle></TrafficHandle>
        <TrafficId>
                <v2TrafficId>
                        <v3TrafficId>
                                <SourceUuid>uuid:1fac1234-31f8-11b4-a222-
08002b34c003</SourceUuid>
                                <DestinationUuid>uuid:2fac1234-31f8-11b4-a222-
08002b34c003</DestinationUuid>
                        </v3TrafficId>
                </v2TrafficId>
        </TrafficId>
        <AvailableOrderedTspecList>
                <Tspec>
                        <TspecIndex>1</TspecIndex>
                        <TrafficClass>AV</TrafficClass>
                        <v2TrafficSpecification>
                                <v3TrafficSpecification>
                                        <RequestedQosType>2</RequestedQosType>
                                        <DataRate>10000000</DataRate>
                                        <E2EMaxDelayHigh>10000</E2EMaxDelayHigh>
                                        <E2EMaxDelayLow>1000</E2EMaxDelayLow>
                                        <E2EMaxJitter>100</E2EMaxJitter>
                                </v3TrafficSpecification>
                        </v2TrafficSpecification>
                </Tspec>
        </AvailableOrderedTspecList>
        <TrafficLeaseTime>10000</TrafficLeaseTime>
        <OptionalPolicyParams>
                <CpName>Amy's CP</CpName>
        </OptionalPolicyParams>
</TrafficDescriptor>
```

The following is an example TrafficDescriptor that a Control Point could create to request prioritized QoS at the *QosManager* Service.

```
<?xml version="1.0" encoding="UTF-8"?>
<TrafficDescriptor
  xmlns="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:prv="http://myPrivate.com"
  xmlns:prv2="http://myPrivate2.com"
  xsi:schemaLocation="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd
    http://www.upnp.org/schemas/qos/TrafficDescriptor-2-20060624.xsd">

    <!-- prv and prv2 are for illustration purpose only to show how vendor specific namespaces
could be added -->
  <TrafficHandle>kiwin1</TrafficHandle>
  <TrafficId>
    <SourceAddress>
      <Ipv4>192.168.1.50</Ipv4>
    </SourceAddress>
    <SourcePort>23</SourcePort>
    <DestinationAddress>
      <Ipv4>192.168.1.53</Ipv4>
    </DestinationAddress>
    <DestinationPort>23</DestinationPort>
    <IpProtocol>1</IpProtocol>
    <v2TrafficId>
      <v3TrafficId>
        <Whatever>whatever</Whatever>
      </v3TrafficId>
      <prv2:MyPrivate2>whatever</prv2:MyPrivate2>
    </v2TrafficId>
    <prv:MyPrivate1>whatever</prv:MyPrivate1>
  </TrafficId>
  <AvailableOrderedTspecList>
    <Tspec>
      <TspecIndex>300</TspecIndex>
      <TrafficClass>AV</TrafficClass>
    </Tspec>
    <Tspec>
      <TspecIndex>2</TspecIndex>
      <TrafficClass>Audio</TrafficClass>
      <v2TrafficSpecification>
        <v3TrafficSpecification>
            <Whatever>whatever</Whatever>
        </v3TrafficSpecification>
        <prv2:MyPrivate2>whatever</prv2:MyPrivate2>
      </v2TrafficSpecification>
      <prv:MyPrivate1>whatever</prv:MyPrivate1>
    </Tspec>
  </AvailableOrderedTspecList>
  <ActiveTspecIndex>300</ActiveTspecIndex>
  <TrafficImportanceNumber>5</TrafficImportanceNumber>
  <v2>
    <v3>
          <Whatever>whatever</Whatever>
    </v3>
    <prv2:MyPrivate2>whatever</prv2:MyPrivate2>
  </v2>
  <prv:MyPrivate1>whatever</prv:MyPrivate1>
  <OptionalPolicyParams>
    <CpName>Amy's CP</CpName>
    <v2OptionalParams>
      <v3OptionalParams>
        <Whatever>whatever</Whatever>
      </v3OptionalParams>
      <prv2:MyPrivate2>whatever</prv2:MyPrivate2>
    </v2OptionalParams>
    <prv:MyPrivate1>whatever</prv:MyPrivate1>
  </OptionalPolicyParams>
</TrafficDescriptor>
```

Different Tspec(s) for the same traffic could be differentiated using traffic class and other TSPEC parameters as shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<TrafficDescriptor
  xmlns="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd
  http://www.upnp.org/schemas/qos/TrafficDescriptor-2-20060624.xsd">
  <TrafficHandle>kiwin1</TrafficHandle>
  <TrafficId>
    <SourceAddress>
      <Ipv4>192.168.1.50</Ipv4>
    </SourceAddress>
    <SourcePort>23</SourcePort>
    <DestinationAddress>
      <Ipv4>192.168.1.53</Ipv4>
    </DestinationAddress>
    <DestinationPort>23</DestinationPort>
    <IpProtocol>1</IpProtocol>
  </TrafficId>
  <AvailableOrderedTspecList>
    <Tspec>
      <TspecIndex>300</TspecIndex>
      <TrafficClass>AV</TrafficClass>
    </Tspec>
    <Tspec>
      <TspecIndex>2</TspecIndex>
      <TrafficClass>Audio</TrafficClass>
    </Tspec>
  </AvailableOrderedTspecList>
  <ActiveTspecIndex>300</ActiveTspecIndex>
  <TrafficImportanceNumber>5</TrafficImportanceNumber>
  <v2>
    <PolicyHolderId>uuid:XYZ-Com-QosPolicyHolder-T001:serviceId:qphT2</PolicyHolderId>
    <PolicyLastModified>2004-11-26T15:03:23-08:00</PolicyLastModified>
    <PolicyModifyingUserName>Amy</PolicyModifyingUserName>
    <PolicyHolderConfigUrl>http://10.0.0.50/ConfigPolicy.html </PolicyHolderConfigUrl>
  </v2>
  <OptionalPolicyParams>
    <CpName>Amy's CP</CpName>
  </OptionalPolicyParams>
</TrafficDescriptor>
```

2.2.3. A_ARG_TYPE_TrafficHandle

This state variable contains a unique identifier associated with a particular instance of TrafficDescriptor, i.e. a particular traffic stream. It is a unique string generated by the *QosManager* and provided to the Control Point in response to the *RequestTrafficQos()* **and** *RequestExtendedTrafficQos()* action. In all subsequent communication between the Control Point and the *QosManager* service, *TrafficHandle* is used to reference a particular traffic stream. The *QosManager* MUST ensure that TrafficHandle is a unique string that identifies one and only one TrafficDescriptor on the network.

2.2.4. A_ARG_TYPE_NumTrafficDescriptors

This state variable contains the number of Traffic Descriptors registered/admitted on the network as reported by the *QosManager* service. This information is returned in response to the *BrowseAllTrafficDescriptors()* action call from the Control Point.

2.2.5. A_ARG_TYPE_NumPolicyHolders

This state variable contains the number of instances of *QosPolicyHolder* service discovered by a given instance of the QosManager service. This variable is returned in response to *RequestTrafficQos()* and *UpdateTrafficQos()* actions (as an output argument) to convey to a Control Point the number of active QosPolicyHolders on the network.

If the value of this variable is not equal to 1, it means that the default policy was used by the *QosManager* to make admission control decisions.

A value of "1" indicates that the policy provided by the Control Point selected *QosPolicyHolder* Service has been used or that exactly one *QosPolicyHolder* was found on the network or the preferred *QosPolicyHolder* on the network has been used. The actual *QosPolicyHolder* Service and its type used by the *QosManager* are identified by the *PolicyHolderConsultedId* and *PolicyHolderConsultedType* parameters in the traffic descriptor.

2.2.6. A_ARG_TYPE_ ListOfTrafficDescriptors

This is a string containing an XML fragment. It contains information describing the ListOfTrafficDescriptors structure. This structure contains a list of traffic descriptors each with the information for a traffic stream.

The XML fragment in this argument MUST validate against the XML schema for ListOfTrafficDescriptors in the XML namespace "`http://www.upnp.org/schemas/ListOfTrafficDescriptors.xsd`" which is located at "`http://www.upnp.org/schemas/qos/ListOfTrafficDescriptors-v3.xsd`".

2.2.6.1. Sample argument XML string

Example 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListOfTrafficDescriptors
xsi:schemaLocation="http://www.upnp.org/schemas/ListOfTrafficDescriptors.xsd
ListOfTrafficDescriptors-v3.xsd"
xmlns="http://www.upnp.org/schemas/ListOfTrafficDescriptors.xsd"
xmlns:n2="http://www.altova.com/samplexml/other-namespace"
xmlns:td2="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<TrafficDescriptor>
 <td2:TrafficHandle>TH1b4-a222-08002b34c0037f921234-723c-11b4</td2:TrafficHandle>
 <td2:TrafficId>
  <SourceAddress>
   <Ipv4>192.168.1.1</Ipv4>
  </SourceAddress>
  <SourcePort>554</SourcePort>
  <DestinationAddress>
   <Ipv4>192.168.1.3</Ipv4>
  </DestinationAddress>
  <DestinationPort>7572</DestinationPort>
  <IpProtocol>17</IpProtocol>
  <v2TrafficId>
   <v3TrafficId>
    <SourceUuid>2fac1234-31f8-11b4-a222-08002b34c003</SourceUuid>
    <DestinationUuid>7f921234-723c-11b4-a222-2fac2b34c003</DestinationUuid>
   </v3TrafficId>
  </v2TrafficId>
 </td2:TrafficId>
 <td2:AvailableOrderedTspecList>
  <td2:Tspec>
   <td2:TspecIndex>1</td2:TspecIndex>
   <AvTransportUri>rtsp://192.168.1.1/Movies/Sample1.mpeg</AvTransportUri>
   <AvTransportInstanceId>0</AvTransportInstanceId>
   <TrafficClass>AV</TrafficClass>
   <v2TrafficSpecification>
    <v3TrafficSpecification>
     <RequestedQosType>2</RequestedQosType>
     <DataRate>15000000</DataRate>
     <PeakDataRate>20000000</PeakDataRate>
     <E2EMaxDelayHigh>10000</E2EMaxDelayHigh>
     <E2EMaxDelayLow>1000</E2EMaxDelayLow>
     <E2EMaxJitter>100</E2EMaxJitter>
     <ServiceType>1</ServiceType>
    </v3TrafficSpecification>
   </v2TrafficSpecification>
  </td2:Tspec>
 </td2:AvailableOrderedTspecList>
 <ActiveTspecIndex>1</ActiveTspecIndex>
 <TrafficImportanceNumber>5</TrafficImportanceNumber>
 <MediaServerConnectionId>8973247048732</MediaServerConnectionId>
 <MediaRendererConnectionId>7492</MediaRendererConnectionId>
 <TrafficLeaseTime>10000</TrafficLeaseTime>
```

```
 <v2>
  <PolicyHolderId>2fac1234-31f8-11b4-a222-08002b34c003:urn:upnp-org:serviceId:QosPolicyHolder-
3a</PolicyHolderId>
  <PolicyLastModified>2006-12-19T16:39:57-08:00</PolicyLastModified>
  <PolicyModifyingUserName>jpaine</PolicyModifyingUserName>
  <PolicyHolderConfigUrl>http://192.168.1.2/QPH.html</PolicyHolderConfigUrl>
  <v3>
   <Critical>0</Critical>
   <PolicyHolderConsultedId>fac1234-31f8-11b4-a222-08002b34c003:urn:upnp-
org:serviceId:QosPolicyHolder-3a</PolicyHolderConsultedId>
   <PolicyHolderConsultedType>1</PolicyHolderConsultedType>
  </v3>
 </v2>
 <OptionalPolicyParams>
  <UserName>jpaine</UserName>
  <CpName>HomeQosPolicyHolder</CpName>
 </OptionalPolicyParams>
</TrafficDescriptor>
</ListOfTrafficDescriptors>
```

Example 2:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListOfTrafficDescriptors
  xmlns="http://www.upnp.org/schemas/ListOfTrafficDescriptors.xsd"
  xmlns:td="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.upnp.org/schemas/ListOfTrafficDescriptors.xsd
  http://www.upnp.org/schemas/qos/ListOfTrafficDescriptors-2-20060624.xsd">
  <TrafficDescriptor>
    <td:TrafficHandle>kiwin1</td:TrafficHandle>
    <td:TrafficId>
      <td:SourceAddress>
        <td:Ipv4>192.168.1.50</td:Ipv4>
      </td:SourceAddress>
      <td:SourcePort>23</td:SourcePort>
      <td:DestinationAddress>
        <td:Ipv4>192.168.1.50</td:Ipv4>
      </td:DestinationAddress>
      <DestinationPort>23</DestinationPort>
      <IpProtocol>1</IpProtocol>
    </td:TrafficId>
    <td:AvailableOrderedTspecList>
      <td:Tspec>
        <td:TspecIndex>300</td:TspecIndex>
        <td:TrafficClass>AV</td:TrafficClass>
      </td:Tspec>
      <td:Tspec>
        <td:TspecIndex>2</td:TspecIndex>
        <td:TrafficClass>Audio</td:TrafficClass>
      </td:Tspec>
    </td:AvailableOrderedTspecList>
    <ActiveTspecIndex>1</ActiveTspecIndex>
    <TrafficImportanceNumber>5</TrafficImportanceNumber>
    <OptionalPolicyParams>
      <CpName>Amy's CP</CpName>
    </OptionalPolicyParams>
  </TrafficDescriptor>
  <TrafficDescriptor>
    <td:TrafficHandle>kiwin2</td:TrafficHandle>
    <td:TrafficId>
      <td:SourceAddress>
        <td:Ipv4>192.168.1.53</td:Ipv4>
      </td:SourceAddress>
      <td:SourcePort>23</td:SourcePort>
      <td:DestinationAddress>
        <td:Ipv4>192.168.1.55</td:Ipv4>
      </td:DestinationAddress>
      <td:DestinationPort>23</td:DestinationPort>
      <td:IpProtocol>1</td:IpProtocol>
    </td:TrafficId>
    <td:AvailableOrderedTspecList>
      <td:Tspec>
        <td:TspecIndex>300</td:TspecIndex>
        <td:TrafficClass>AV</td:TrafficClass>
      </td:Tspec>
```

```
      <td:Tspec>
        <td:TspecIndex>2</td:TspecIndex>
        <td:TrafficClass>Audio</td:TrafficClass>
      </td:Tspec>
    </td:AvailableOrderedTspecList>
    <ActiveTspecIndex>1</ActiveTspecIndex>
    <TrafficImportanceNumber>5</TrafficImportanceNumber>
    <OptionalPolicyParams>
      <CpName>Amy's CP</CpName>
    </OptionalPolicyParams>
  </TrafficDescriptor>
</ListOfTrafficDescriptors>
```

2.2.7. A_ARG_TYPE_QmCapabilities

This is a string containing an XML fragment. It contains information describing *QosManager* Service capabilities. The XML fragment in this argument MUST validate against the XML schema for *QmCapabilities* in the XML namespace "`urn:schemas-upnp-org:qos:QmCapabilities`" which is located at "`http://www.upnp.org/schemas/qos/QmCapabilities-v3.xsd`".

2.2.7.1. Description of fields in the A_ARG_TYPE_QmCapabilities structure

ReportBlockingStreams: This **boolean** field is a required field. The *ReportBlockingStreams* field indicates whether information about blocking streams will be reported to a Control Point or not.

Preemption: This **boolean** field is a required field. The *Preemption* field indicates whether this *QosManager* performs Preemption.

2.2.7.2. Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2007 rel. 3 sp1 (http://www.altova.com)-->
<QmCapabilities xsi:schemaLocation="urn:schemas-upnp-org:qos:QmCapabilities QmCapabilities-v3-
20070510.xsd" xmlns="urn:schemas-upnp-org:qos:QmCapabilities"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <PreemptionCapabilities>
                <ReportBlockingStreams>1</ReportBlockingStreams>
                <Preemption>1</Preemption>
        </PreemptionCapabilities>
</QmCapabilities>
```

2.2.8. A_ARG_TYPE_ExtendedTrafficQosInfo

This is a string containing an XML fragment. It contains information describing *ExtendedTrafficQosInfo* structure.

The XML fragment in this argument MUST validate against the XML schema for *ExtendedTrafficQosInfo* in the XML namespace "`urn:schemas-upnp-org:qos:ExtendedTrafficQosInfo`" which is located at "`http://www.upnp.org/schemas/qos/ExtendedTrafficQosInfo-v3.xsd`"

This structure provides information about a list of existing streams blocking the admission of a new traffic stream on a given QoS Segment. The information contained in this structure is obtained by the *QosManager* from *QosDevice* Services residing on the failed QoS Segments on the path of the requested stream.

The information for an individual blocking stream consists of layer-2 stream information such as QosSegmentId, Layer2StreamId, and UPnP-QoS specific information such as TrafficDescriptor and the UserImportanceNumber, if available.

2.2.8.1. Description of fields in the A_ARG_TYPE_ExtendedTrafficQosInfo structure

UINOfCurrentStream: This is a required field is of type **ui4**. It represents the *UserImportanceNumber* of a traffic stream that cannot be admitted due to Blocking Streams listed in this structure. It's value ranges from 0 to 255.

QosSegmentId: The value is of type **string** and uniquely identifies a failed QoS Segment for a blocking stream.

TrafficDescriptor: This is an optional field. It provides the TrafficDescriptor of a Blocking Stream on a given QoS Segment.

UserImportanceNumber: This is an optional field. It provides the *UserImportanceNumber* of a Blocking Stream.

Layer2StreamId: This is a required field. This indicates layer-2 identifier of a blocking stream. This is included in the return argument of *QD:AdmitTrafficQos* or *QD:UpdateTrafficQos* action upon failure.

2.2.8.2. Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2008 sp1 (http://www.altova.com)-->
<ExtendedTrafficQosInfo xsi:schemaLocation="urn:schemas-upnp-org:qos:ExtendedTrafficQosInfo
ExtendedTrafficQosInfo-v3.xsd" xmlns="urn:schemas-upnp-org:qos:ExtendedTrafficQosInfo"
xmlns:n2="http://www.altova.com/samplexml/other-namespace"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<UINOfCurrentStream>200</UINOfCurrentStream>
<BlockingStreamsPerSegment>
  <QosSegmentId>174957c7aff8e2a0d</QosSegmentId>
  <BlockingStreamInfo>
   <TrafficDescriptor>
    <TrafficHandle>TH1b4-a222-08002b34c0037f921234-723c-11b4</TrafficHandle>
    <TrafficId>
     <SourceAddress>
      <Ipv4>192.168.1.1</Ipv4>
     </SourceAddress>
     <SourcePort>554</SourcePort>
     <DestinationAddress>
      <Ipv4>192.168.1.3</Ipv4>
     </DestinationAddress>
     <DestinationPort>7572</DestinationPort>
     <IpProtocol>17</IpProtocol>
     <v2TrafficId>
      <v3TrafficId>
       <SourceUuid>2fac1234-31f8-11b4-a222-08002b34c003</SourceUuid>
        <DestinationUuid>7f921234-723c-11b4-a222-2fac2b34c003</DestinationUuid>
      </v3TrafficId>
     </v2TrafficId>
    </TrafficId>
    <AvailableOrderedTspecList>
    <Tspec>
     <TspecIndex>1</TspecIndex>
        <AvTransportUri>rtsp://192.168.1.1/Movies/Sample1.mpeg</AvTransportUri>
        <AvTransportInstanceId>0</AvTransportInstanceId>
        <TrafficClass>AV</TrafficClass>
        <v2TrafficSpecification>
         <v3TrafficSpecification>
          <RequestedQosType>2</RequestedQosType>
          <DataRate>15000000</DataRate>
          <PeakDataRate>20000000</PeakDataRate>
          <E2EMaxDelayHigh>10000</E2EMaxDelayHigh>
          <E2EMaxDelayLow>1000</E2EMaxDelayLow>
          <E2EMaxJitter>100</E2EMaxJitter>
          <QosSegmentSpecificParameters>
           <InterfaceId>String</InterfaceId>
           <QosSegmentId>174957c7aff8e2a0d</QosSegmentId>
           <QosSegmentMaxDelayHigh>5000</QosSegmentMaxDelayHigh>
           <QosSegmentMaxDelayLow>500</QosSegmentMaxDelayLow>
           <QosSegmentMaxJitter>75</QosSegmentMaxJitter>
          </QosSegmentSpecificParameters>
          <ServiceType>true</ServiceType>
         </v3TrafficSpecification>
        </v2TrafficSpecification>
    </Tspec>
```

```
   </AvailableOrderedTspecList>
   <ActiveTspecIndex>1</ActiveTspecIndex>
   <TrafficImportanceNumber>5</TrafficImportanceNumber>
   <MediaServerConnectionId>8973247048732</MediaServerConnectionId>
   <MediaRendererConnectionId>7492</MediaRendererConnectionId>
   <TrafficLeaseTime>10000</TrafficLeaseTime>
   <v2>
    <PolicyHolderId>2fac1234-31f8-11b4-a222-08002b34c003:urn:upnp-
org:serviceId:QosPolicyHolder-3a</PolicyHolderId>
    <PolicyLastModified>2006-12-19T16:39:57-08:00</PolicyLastModified>
    <PolicyModifyingUserName>jpaine</PolicyModifyingUserName>
    <PolicyHolderConfigUrl>http://192.168.1.2/QPH.html</PolicyHolderConfigUrl>
    <v3>
     <Critical>0</Critical>
     <PolicyHolderConsultedId>2fac1234-31f8-11b4-a222-08002b34c003:urn:upnp-
org:serviceId:QosPolicyHolder-3a</PolicyHolderConsultedId>
     <PolicyHolderConsultedType>3</PolicyHolderConsultedType>
    </v3>
   </v2>
   <OptionalPolicyParams>
    <UserName>jpaine</UserName>
    <CpName>HomeQosPolicyHolder</CpName>
   </OptionalPolicyParams>
  </TrafficDescriptor>
  <UserImportanceNumber>10</UserImportanceNumber>

<Layer2StreamId>0c14cd560c14cd560c14cd560c14cd560c14cd560c14cd560c14cd560c14cd56</Layer2Stream
Id>
  </BlockingStreamInfo>
 </BlockingStreamsPerSegment>
</ExtendedTrafficQosInfo>
```

2.2.9. A_ARG_TYPE_ResultedQosType

This is a non-negative integer field that indicates the type of Quality of Service established for a traffic stream by the *QoS Manager* as a result of *RequestExtendedTrafficQos()* or *UpdateExtendedTrafficQos()* actions.

Table 2-5: Meaning of *ResultedQosType* parameter

Value	Meaning
0	Prioritized QoS The traffic stream is successfully set up using prioritized QoS for every QoS segment it traverses This includes, in particular, prioritized QoS setup for QoS segments that support parameterized QoS.
1	Hybrid QoS The traffic stream is successfully admitted on every parameterized QoS segment it traverses and is set up using prioritized QoS on every prioritized QoS segment it traverses.
2	Parameterized QoS This traffic stream is successfully admitted on every parameterized QoS segments it traverses and the traffic stream does not traverse any non-parameterized QoS segment.
3 and higher	Reserved

2.2.10. Relationships Between State Variables

None

2.3. Eventing and Moderation

Table 2-6: Event Moderation

Variable Name	Evented	Moderated Event	Max Event Rate[1]	Logical Combination	Min Delta per Event[2]
A_ARG_TYPE_TrafficDescriptor	*NO*	*n/a*	*n/a*	*n/a*	*n/a*
A_ARG_TYPE_TrafficHandle	*NO*	*n/a*	*n/a*	*n/a*	*n/a*
A_ARG_TYPE_NumTrafficDescriptors	*NO*	*n/a*	*n/a*	*n/a*	*n/a*
A_ARG_TYPE_NumPolicyHolders	*NO*	*n/a*	*n/a*	*n/a*	*n/a*
A_ARG_TYPE_ListOfTrafficDescriptors	*NO*	*n/a*	*n/a*	*n/a*	*n/a*
A_ARG_TYPE_QmCapabilities	*NO*	*n/a*	*n/a*	*n/a*	*n/a*
A_ARG_TYPE_ExtendedTrafficQosInfo	*NO*	*n/a*	*n/a*	*n/a*	*n/a*
A_ARG_TYPE_ ResultedQosType	*NO*	*n/a*	*n/a*	*n/a*	*n/a*

[1] Determined by N, where Rate = (Event)/(N secs).
[2] (N) * (allowedValueRange Step).

2.3.1. Event Model

This service does not expose any eventing state variables.

2.4. Actions

Immediately following this table is detailed information about these actions, including short descriptions of the actions, the effects of the actions on state variables, and error codes defined by the actions.

Table 2-7: Actions

Name	Req. or Opt. [1]
RequestTrafficQos()	R
UpdateTrafficQos()	R
ReleaseTrafficQos()	R
BrowseAllTrafficDescriptors()	R
GetQmCapabilities()	R
RequestExtendedTrafficQos()	O
UpdateExtendedTrafficQos()	O

[1] R = Required, O = Optional, X = Non-standard.

2.4.1 RequestTrafficQos()

A Control Point invokes this action for setting up QoS for a particular traffic stream. This action does not support optional extended capabilities such as preemption, reporting of blocking streams, etc.

The Control Point uses the *RequestTrafficQos()* action exposed by the *QosManager* Service to establish prioritized, parameterized or hybrid QoS for a stream. The Control Point puts the *InitialTrafficDescriptor* as an input argument in the *RequestTrafficQos()* action. The type of QoS established for the stream is determined by the absence or presence and value of the *RequestedQosType* parameter in the *InitialTrafficDescriptor*.

If the Control Point does not include the *RequestedQosType* parameter in the *InitialTrafficDescriptor* (e.g., legacy 2.0/1.0 CP), by default, the *QoS Manager* attempts to establish prioritized QoS on all the QoS Segments on the path of the stream. If the value of *RequestedQosType* parameter is set to "0" (i.e., prioritized-only), the *QoS Manager* attempts to establish prioritized QoS on all the QoS Segments on the path of the stream. If the value of the *RequestedQosType* parameter is set to "1" (i.e., Hybrid), the *QoS Manager* attempts to establish parameterized QoS on the QoS Segments that support parameterized QoS and attempts to establish prioritized QoS on the QoS Segments that only support prioritized QoS. If the value of the *RequestedQosType* parameter is set to "2" (i.e., parameterized-only), the *QoS Manager* attempts to establish parameterized QoS on all the QoS Segments on the path of the stream.

2.4.1.1. Arguments

Table 2-8: Arguments for *RequestTrafficQos()*

Argument	Direction	relatedStateVariable
InitialTrafficDescriptor	In	A_ARG_TYPE_TrafficDescriptor
TrafficHandle	Out	A_ARG_TYPE_TrafficHandle
NumPolicyHolders	Out	A_ARG_TYPE_NumPolicyHolders
UpdatedTrafficDescriptor	Out	A_ARG_TYPE_TrafficDescriptor

2.4.1.2. Service requirements

2.4.1.2.1. Base Requirements

The following generic requirements apply irrespective of the absence or presence and the value of the *RequestedQosType* parameter in the *InitialTrafficDescriptor* input argument:

- If a Control Point supplies *TrafficImportanceNumber* in TrafficDescriptor to the *QosManager* Service when calling the *RequestTrafficQos()* action, the *QosManager* Service MUST return an error (Error Code 721).
- If a Control Point supplies *ActiveTspecIndex* in TrafficDescriptor to *QosManager* Service when calling the *RequestTrafficQos()* action, the *QosManager* Service MUST return an error (Error Code 722).
- The *QosManager* Service MUST include a valid *ActiveTspecIndex* value in the TrafficDescriptor when it returns a non error value in response to *RequestTrafficQos()*.
- If a Control Point supplies a non-null *TrafficHandle* in the *RequestTrafficQos()* action of *QosManager* Service, the *QosManager* Service MUST return error code 701.
- If a Control Point supplies *PolicyLastModified*, *PolicyModifyingUserName*, or *PolicyHolderConfigUrl* when calling the *RequestTrafficQos()* action, the *QosManager* Service MUST return error code 715.

- If a Control Point does not supply a unique *TspecIndex* for every Tspec within an *AvailableOrderedTspecList*, then the *QosManager* Service MUST return the error 725.

- If the *QosManager* consults a *QosPolicyHolder* Service, the *QosManager* Service MUST include the TrafficPolicy parameters received from *QPH:GetTrafficPolicy()* or *QPH:GetListOfTrafficPolicies()* action in the TrafficDescriptor when successfully returning the *UpdatedTrafficDescriptor*. The *QosManager* Service MUST always include *QosPolicyHolderId*.

- If the *QoS Manager* is successful in establishing QoS for a traffic stream on the entire path (i.e. *QD:SetupTrafficQos()* and/or *QD:AdmitTrafficQos()* actions are successfully executed on all the *QosDevice* Service on the path), the *QosManager* Service MUST return the admitted traffic descriptor to the Control Point in the *UpdatedTrafficDescriptor* output argument of the *RequestTrafficQos()* action.

- The *QosManager* Service MUST populate all the fields of the *TrafficId* in the *UpdatedTrafficDescriptor*.

- If the *QoS Manager* obtains the *MaxCommittedDelay* for every QoS Segment on the path, then the *QosManager* Service MUST populate *E2EMaxDelayHigh* in the *UpdatedTrafficDescriptor*.

- If the *QoS Manager* obtains the *MaxCommittedDelayJitter* for every QoS Segment on the path, then the *QosManager* Service MUST populate the *E2EMaxJitter* in the *UpdatedTrafficDescriptor*.

- The *QosManager* Service MUST NOT return *QosSegmentSpecificParameter* in the *UpdatedTrafficDescriptor*.

- If the *QosManager* consults a *QosPolicyHolder* Service, the *QosManager* Service MUST return the value of "1" for the *NumPolicyHolders* output argument if it uses the preferred *QosPolicyHolder* Service OR the *QosPolicyHolder* Service supplied by the Control Point OR a single available *QosPolicyHolder* Service on the network. The actual *QosPolicyHolder* Service and its type used by the *QosManager* Service are identified by the *PolicyHolderConsultedId* and *PolicyHolderConsultedType* parameters in the *UpdatedTrafficDescriptor* output argument.

- If the QosManager doesn't consult a *QosPolicyHolder* Service (e.g. in case of RequestedQosType = "2" or when the *QoS Manager* uses default policies for RequestedQosType = "0" and "1"), the *QosManager* Service MUST return the actual number of *QosPolicyHolder* Services discovered on the network.

2.4.1.2.2. RequestedQosType Specific Requirements

The following requirements apply when *RequestedQosType* parameter is present in the *InitialTrafficDescriptor* input parameter:

- For the *RequestedQosType* parameter with the value of "1" or "2", if the *QoS Manager* is unable to successfully admit the traffic stream on all the 3.0 *QosDevice* Services on the path (with or without retries), the *QosManager* Service MUST return Error Code 765 to the Control Point.

- For the *RequestedQosType* parameter with the value of "2", if the *QoS Manager* identifies that some of the QoS Segments on the path do not support parameterized QoS and no less preferred TSPEC is supplied, the *QosManager* Service MUST return error to the CP in response to *RequestTrafficQos()* action with error code of 782.

- If a Control Point supplies the *RequestedQosType* parameter with the value greater than "2", the *QosManager* Service MUST return error with Error Code 718 to the Control Point.

- For the *RequestedQosType* parameter with the value of "1" or "2", if a Control Point does not supply a value for the *TrafficLeaseTime* field in the *InitialTrafficDescriptor* input argument, the *QosManager* Service MUST return error with Error Code 791 to the Control Point.

- For the *RequestedQosType* parameter with the value of "1" or "2", if a Control Point supplies the value for the *E2EMaxDelayLow* parameter but does NOT supply the value for the *E2EMaxDelayHigh* in the input TSPEC parameters, the *QosManager* Service MUST return error with Error Code 792 to the Control Point.

2.4.1.3. Control Point requirements when calling the action

2.4.1.3.1. Base Requirements

The following generic requirements apply irrespective of the absence or presence and value of *RequestedQosType* parameter in the *InitialTrafficDescriptor* input parameter:

- A Control Point MUST supply a *TrafficHandle* element with a null value (i.e. empty string) to the *RequestTrafficQos()* action.
- A Control Point MUST supply a *TrafficId* structure which contains at least either *SourceAddress* or *SourceUuid* and either *DestinationAddress* or *DestinationUuid*.
- A UPnP AV Control Point, when invoking the *QosManager* Service, MUST supply the *AVTransportURI*, *AVTransportInstanceId*, *MediaServerConnectionId* and *MediaRendererConnectionId*.
- A Control Point MUST NOT supply *TrafficImportanceNumber* & *UserImportanceNumber* in *InitialTrafficDescriptor* when calling the *RequestTrafficQos()* action.
- A Control Point MUST supply a unique *TspecIndex* for every Tspec within an *AvailableOrderedTspecList*.
- A Control Point MUST NOT supply an *ActiveTspecIndex* when invoking *RequestTrafficQos()*.
- If a Control Point supplies a *QosBoundarySourceAddress,* it MUST be a valid IP Address. The 0.0.0.0 option for the "don't care" address does not apply to *QosBoundarySourceAddress*.
- If a Control Point supplies a *QosBoundaryDestinationAddress* it MUST be a valid IP Address. The 0.0.0.0 option for the "don't care" address does not apply to *QosBoundaryDestinationAddress*.
- Whenever a Control Point specifies a *PolicyHolderId* it MUST specify a valid Id for a PolicyHolder as defined in the *QosPolicyHolder* service definition. [POLICY HOLDER].
- A Control Point MUST NOT supply *PolicyLastModified*, *PolicyModifyingUserName*, *PolicyHolderConfigUrl* as a part of *InitialTrafficDescriptor*.

2.4.1.3.2. RequestedQosType Specific Requirements

The following requirements apply when *RequestedQosType* parameter is present in the *InitialTrafficDescriptor* input parameter:

- A Control Point MUST NOT supply a value greater than "2" for the *RequestedQosType parameter.*
- For the *RequestedQosType* parameter with the value of "1" or "2", a Control Point MUST supply a value for the *TrafficLeaseTime* field in the *InitialTrafficDescriptor* input argument.
- For the *RequestedQosType* parameter with the value of "1" or "2", if a Control Point supplies a value for the *E2EMaxDelayLow* as one of the input TSPEC parameters, then the Control Point MUST also supply the value for the *E2EMaxDelayHigh* parameters in the TSPEC.

2.4.1.4. QoS Manager Requirements

2.4.1.4.1. Base Requirements

The following generic requirements apply irrespective of the absence or presence and value of *RequestedQosType* parameter in the *InitialTrafficDescriptor* input parameter:

The *QoS Manager* MUST NOT supply any of the TrafficPolicy elements in the TrafficDescriptor when calling the **QPH:GetTrafficPolicy** action.

When a Control Point calls the *RequestTrafficQos()* action and the *QoS Manager* subsequently invokes actions with a TrafficDescriptor as an argument, the *QoS Manager* MUST NOT modify those elements that MUST equal the original TrafficDescriptor according to Table 2-9.

Table 2-9 Elements that MUST equal those of the original TrafficDescriptor

Parameter Name	Must equal original TrafficDescriptor?	Comments
TrafficHandle	No	
TrafficId	Not applicable	See sub–elements below
. SourceAddress	Yes, if supplied	If not supplied by the Control Point, QM obtains it from the QD
. SourcePort	Yes, if supplied	If not supplied by the Control Point, QM obtains it from the QD
. DestinationAddress	Yes, if supplied	If not supplied by the Control Point, QM obtains it from the QD
. DestinationPort	Yes, if supplied	If not supplied by the Control Point, QM obtains it from the QD
. IpProtocol	Yes, if supplied	If not supplied by the Control Point, QM obtains it from the QD
. v2TrafficId	Not applicable	See sub-elements below
. . v3TrafficId	Not applicable	See sub-elements below
. . . SourceUuid	Yes, if supplied	
. . . DestinationUuid	Yes, if supplied	
AvailableOrderedTspecList	Not applicable	See sub-elements below
. Tspec	Yes	
. . TspecIndex	Yes	
. . AvTransportUri	Yes	
. . AvTransportInstanceId	Yes	
. . TrafficClass	Yes	
. . v2TrafficSpecification	Not applicable	See sub-elements below
. . . v3TrafficSpecification	Not applicable	See sub-elements below
. . . . RequestedQosType	Yes, if supplied	

Parameter Name	Must equal original TrafficDescriptor?	Comments
. . . . DataRate	Yes	
. . . . MaxBurstSize	Yes	
. . . . MinServiceRate	Yes	
. . . . ReservedServiceRate	Yes	
. . . . TimeUnit	Yes	
. . . . MaxPacketSize	Yes	
. . . . E2EMaxDelayHigh	Yes	
. . . . E2EMaxDelayLow	Yes	
. . . . E2EMaxJitter	Yes	
. . . . QosSegmentSpecificParameters	Not applicable	See sub-elements below
. . . . . InterfaceId	Not applicable	
. . . . . QosSegmentId	Not applicable	
. . . . . QosSegmentMaxDelayHigh	Not applicable	
. . . . . QosSegmentMaxDelayLow	Not applicable	
. . . . . QosSegmentMaxJitter	Not applicable	
. . . . MaxServiceInterval	Yes	
. . . . MinServiceInterval	Yes	
. . . . LossSensitivity	Yes	
. . . . ServiceType	Yes	
ActiveTspecIndex	No	
TrafficImportanceNumber	No	
QosBoundarySourceAddress	Yes	
QosBoundaryDestinationAddress	Yes	
MediaServerConnectionId	Yes	
MediaRendererConnectionId	Yes	
TrafficLeaseTime	Yes	
PolicyHolderId	No	
PolicyLastModified	No	
PolicyModifyingUserName	No	
PolicyHolderConfigUrl	No	
QosBoundarySourceUuid	Yes	
QosBoundaryDestinationUuid	Yes	
Critical	Yes	

Parameter Name	Must equal original TrafficDescriptor?	Comments
PolicyHolderConsultedId	No	
PolicyHolderConsultedType	No	
OptionalPolicyParams	Not applicable	See sub-elements below
. UserName	Yes	
. VendorApplicationName	Yes	
. PortName	Yes	
. ServiceProviderServiceName	Yes	
. CpName	Yes	

2.4.1.4.2. QoS Policy Retrieval Requirements for establishing Prioritized QoS

If the *QoS Manager* fails to discover any *QosPolicyHolder* service, the *QoS Manager* MUST perform the following steps:

- The *QoS Manager* MUST use the default policy of first-come-first serve admission control
- The *QoS Manger* MUST use the default priorities based on IEEE 802.1D Annex G as described below in Table 2-5.
- The *QoS Manager* MUST use default value of "Enabled" for the AdmissionPolicy.

The following table describes how the *QoS Manager* derives default priorities to be used in the TrafficImportanceNumber field. The *QoS Manager* takes the value of Traffic Class from the TSPEC found inside the Traffic Descriptor structure, and looks up the corresponding default priority from the following table:

Table 2-10 Default Priorities used by the *QoS Manager* based on 802.1D (Annex G)

Traffic Class	802.1D (Annex G) Traffic Type	Default Priority (Traffic Importance Number)
Network Control	NC	7
Streaming Control	NC	7
Voice	VO	6
Gaming	VO	6
AV	VI	5
Audio	VI	5
Image	EE	3
Data	BE	0
Background	BK	1
Other	BE	0

When a Control Point calls the *RequestTrafficQos()* action, if a *QoS Manager* discovers one or more *QosPolicyHolder* Services on the network the *QoS Manager* MUST follow the following steps:

- If the *QosPolicyHolder* Service identified in the *PolicyHolderId* parameter of the input traffic descriptor is available, the *QoS Manager* MUST use the *QPH:GetTrafficPolicy()* action to retrieve the TrafficPolicy from that *QosPolicyHolder* service. The *QoS Manager* MUST perform admission control, as defined in UPnP-QoS v2, if the *AdmissionPolicy* output parameter of the *QPH:GetTrafficPolicy()* is *Enabled*. If the *QosPolicyHolder* Service identified in the *PolicyHolderId* is not available, the *QosManager* responds to *RequestTrafficQos()* or *UpdateTrafficQos()* with error 780.

- If a *PolicyHolderId* is not provided in the *PolicyHolderId* parameter of the input traffic descriptor, the *QoS Manger* MUST identify if there is a preferred *QosPolicyHolder* Service on the network as described in section 2.4.1.4.3.

- If the *QoS Manager* determines that there is a preferred *QosPolicyHolder* Service selected and available on the network, the *QoS Manager* MUST invoke *QPH:GetTrafficPolicy()* on the preferred *QosPolicyHolder* Service. The *QoS Manager* MUST set the *PolicyHolderConsultedType* in the TrafficDescriptor to "3" (Preferred) and MUST populate the UDN+ServiceId of the preferred *QosPolicyHolder* in the PolicyHolderConsultedId parameter. If the preferred *QosPolicyHolder* isn't available, the *QoS Manager* MUST use default policy of first-come-first serve admission control with default priorities based on IEEE 802.1D Annex G. The *QoS Manager* MUST set the *PolicyHolderConsultedType* in the TrafficDescriptor to "0" (Default) and MUST set *PolicyHolderConsultedId* parameter to NULL.

- If the *QoS Manager* determines that no preferred *QosPolicyHolder* Service is selected on the network and there is a single *QosPolicyHolder* on the network, the *QoS Manager* MUST invoke *QPH:GetTrafficPolicy()* on that *QosPolicyHolder* Service. The *QoS Manager* MUST perform admission control, as defined in UPnP-QoS v2, if the *AdmissionPolicy* output parameter of the *QPH:GetTrafficPolicy()* is *Enabled*. The *QoS Manager* MUST set the *PolicyHolderConsultedType* in the TrafficDescriptor to "1" (Single-Available) and MUST set *PolicyHolderConsultedId* parameter to the the UDN+ServiceId of the single *QosPolicyHolder* Service available on the network.

- If the *QoS Manager* determines that no preferred *QosPolicyHolder* Service is selected on the network and there are multiple *QosPolicyHolder* Services on the network, the *QoS Manager* MUST use default policy of first-come-first serve admission control with default priorities based on IEEE 802.1D Annex G as specified above in Section 2.4.1.4.2 The *QoS Manager* MUST use the default value of "Enabled" for the AdmissionPolicy. The *QoS Manager* MUST set the PolicyHolderConsultedType in the TrafficDescriptor to "0" (Default) and MUST set *PolicyHolderConsultedId* parameter to NULL.

2.4.1.4.3. QoS Policy Retrieval Requirements for Parameterized and Hybrid QoS

When the QoS Manager needs to access QoS policies for a Hybrid or Parameterized stream, it MUST perform the following steps:

The *QoS Manager* invokes the *QD:SetPreferredQph()* action on all the *QosDevice* Services that implement this action to retrieve the currently stored values of *PreferredQphId* and the *QphPreferenceCount* in those devices. The *QoS Manager* determines which *QphPreferenceCount* is the highest of all those retrieved from *QosDevice* Services and uses that *QphPreferenceCount* to identify the preferred *QosPolicyHolder* Service. The *QosManager* MUST use the *QosPolicyHolder* Service with the highest *QphPreferenceCount* as the preferred *QosPolicyHolder* service.

If the *PreferredQphId* value is NULL, then the *QoS Manager* determines that the user did not prefer any *QosPolicyHolder* Service on the network.

If the *QoS Manager* determines that there is a preferred *QosPolicyHolder* Service selected and available on the network, for RequestedQosType = "1", the *QoS Manager* MUST invoke *QPH:GetTrafficPolicy()* action on that preferred *QosPolicyHolder* Service. When performing "ReportBlockingStreans" or "Preemption" operations for *RequestedQosType* = "1" or "2", the *QoS Manager* MUST invoke *QPH:GetListOfTrafficPolicies()* on the preferred *QosPolicyHolder* Service. The *QoS Manager* MUST set the *PolicyHolderConsultedType* in the TrafficDescriptor to "3" (Preferred) and MUST populate the UDN+ServiceId of the preferred *QosPolicyHolder* in the *PolicyHolderConsultedId* parameter. If the preferred *QosPolicyHolder* isn't available, the *QoS Manager* MUST use default policy of first-come-first serve admission control with default priorities based on IEEE 802.1D Annex G. The *QoS Manager* MUST set the *PolicyHolderConsultedType* in the TrafficDescriptor to "0" (Default) and MUST set *PolicyHolderConsultedId* parameter to NULL.

If the *QoS Manager* determines that no preferred *QosPolicyHolder* Service is selected on the network and there is a single *QosPolicyHolder* on the network, the *QoS Manager* MUST invoke on that *QosPolicyHolder* Service *QPH:GetTrafficPolicy()* action for *RequestedQosType = "1"*. When performing "ReportBlockingStreans" or "Preemption" operations for *RequestedQosType* = "1" or "2", the *QoS Manager* MUST invoke *QPH:GetListOfTrafficPolicies()* on the single available *QosPolicyHolder* Service. The *QoS Manager* MUST set the *PolicyHolderConsultedType* in the TrafficDescriptor to "1" (Single-Available) and MUST set *PolicyHolderConsultedId* parameter to the the UDN+ServiceId of the single *QosPolicyHolder* Service available on the network.

If the *QoS Manager* determines that no preferred *QosPolicyHolder* Service is selected on the network and there are multiple *QosPolicyHolder* Services on the network, the *QoS Manager* identifies if the *QosPolicyHolder* supplied by the Control Point (in the *PolicyHolderId* parameter of the traffic descriptor) is available on the network. If the Control Point specified *QosPolicyHolder* is available, the *QoS Manager* MUST invoke on that *QosPolicyHolder* service *QPH:GetTrafficPolicy()* action for RequestedQosType = "1". When performing "ReportBlockingStreans" or "Preemption" operations for *RequestedQosType* = "1" or "2", the *QoS Manager* MUST invoke *QPH:GetListOfTrafficPolicies()* on the CP supplied *QosPolicyHolder* Service. The *QoS Manager* MUST set the *PolicyHolderConsultedType* in the TrafficDescriptor to "2" (CP Supplied) and MUST set *PolicyHolderConsultedId* parameter to the UDN+ServiceId of the *QosPolicyHolder* supplied by the Control Point. If the *QosPolicyHolder* identified by the *PolicyHolderId* is not available OR the *PolicyHolderId* parameter is not provided, the *QoS Manager* MUST use default policy of first-come-first serve admission control with default priorities based on IEEE 802.1D Annex G as specified above in Section 2.4.1.4.2 The *QoS Manager* MUST use the default value of "Enabled" for the *AdmissionPolicy*. The *QoS Manager* MUST set the *PolicyHolderConsultedType* in the TrafficDescriptor to "0" (Default) and MUST set *PolicyHolderConsultedId* parameter to NULL.

2.4.1.4.4. Prioritized QoS Establishment Requirements

Upon receiving *RequestTrafficQos()* action with the *RequestedQosType* parameter in the *InitialTrafficDescriptor* either absent or of value "0", the *QoS Manager* MUST perform the following steps to establish Prioritized QoS for the stream:

The *QoS Manager* MUST obtain QoS traffic policies as specified in Section 2.4.1.4.2

The *QoS Manager* MAY invoke *QD:GetPathInformation()* action on *QosDevice* service instances on the network to determine which intermediate devices are on the path from the source to the sink. Some UPnP *QosDevice* service instances may also expose the *QD:GetQosDeviceInfo()* action. The *QoS Manager* MAY invoke this action on the source and/or sink device to find out the IP addresses, port number and protocol information associated with that particular Traffic Descriptor, if they are not supplied by the CP. It is possible that some *QosDevice* services on the path are v3 compliant and some are v2 or older. Depending on their version, the *QoS Manager* MUST invoke the *QD:GetQosState()* action on v2 *QosDevice* Sevices and/or *QD:GetExtendedQosState()* action on v3 *QosDevice* Services on the path of the traffic stream to obtain dynamic status about devices. The *QoS Manager* MAY also invoke *QD:GetQosDeviceCapabilities()* action on v2 *QosDevice* Services.

QosStateId is a state variable defined in the *QosDevice* that identifies the state of the device when it was queried by a *QoS Manager* with *QD:GetQosState()* action. *QosDevice* returns *QosStateId* in response to *QD:GetQosState().* Subsequently, the *QoS Manager* repeats this string as input argument to *QD:SetupTrafficQos().* Please see *QosDevice* Service [QOS DEVICE] for more details.

The *QoS Manager* MUST establish QoS for the most preferred of the TSPECs that are admissible on all the *QosDevice* Services on the path from those supplied by the CP in the *InitialTrafficDescriptor.* The *QoS Manager* MUST NOT leave any unused allocated resources on the *QosDevice* Services.

For v2 *QosDevice* Services on the path, the QosManager MUST invoke *QD:SetupTrafficQos()* action. For v3 *QosDevice* Services on the path, the *QoS Manager* MUST invoke either *QD:AdmitTrafficQos()* or *QD:SetupTrafficQos()* action. Invocation of *QD:AdmitTrafficQos()* action is recommended.

If *QD:SetupTrafficQos()* and/or *QD:AdmitTrafficQos()* actions are successfully executed on all the *QosDevice* Services involved, then the *UpdatedTrafficDescriptor* created by the *QoS Manager* is returned to the Control Point as an output argument to the *RequestTrafficQos()* action. The *QoS Manager* MUST set the value of *ActiveTspecIndex* in the *UpdatedTrafficDescriptor* to the index of the TSPEC for which the QoS is established.

If *QD:SetupTrafficQos()* and/or *QD:AdmitTrafficQos()* action fails on any one of the *QosDevice* Services, even after retries (if performed), the *QoS Manager* MUST return Error Code 765 in response to the *RequestTrafficQos()* action.

PathInformation state variable of *QosDevice* Service is mandatory and it is evented with moderation. Any time there is a change in path information the *QosDevice* issues an event and sends the updated *PathInformation* variable in the body of the event. See [QOS DEVICE] for more details. The *QosManager* Service MAY subscribe to these events. Upon receiving an event, the *QosManager* Service MAY take an action that it deems necessary e.g. recomputing the topology.

2.4.1.4.5. Parameterized and Hybrid QoS Establishment Requirements

Upon receiving *RequestTrafficQos()* action with the *RequestedQosType* parameter in the *InitialTrafficDescriptor* of either "1" or "2", the *QoS Manager* MUST perform the following steps to establish QoS for the stream:

If the *RequestedQosType* parameter is set to "1", the *QoS Manager* MUST obtain QoS traffic policies as specified in Section 2.4.1.4.3.

If the *RequestedQosType* parameter is set to "2", the *QoS Manager* MAY obtain QoS traffic policies as specified in Section 2.4.1.4.3.

The *QoS Manager* MUST determine which intermediate *QosDevice* Services are on the path from the source to the sink.. The *QoS Manager* MAY invoke *QD:GetPathInformation()* action on *QosDevice* service instances on the network to determine this or may have this information a priori. It is possible that some of the *QosDevice* services on the path may not be able to support parameterized-based QoS (e.g. v2 or v1 *QosDevice* Services).

The *QoS Manager* MUST invoke *QD:GetQosState()* action on 2.0 or older *QosDevice* Services on the path and MUST invoke *QD:GetExtendedQosState()* on 3.0 *QosDevice* services on the path to obtain the dynamic status of devices. The *QoS Manager* MAY also invoke *QD:GetQosDeviceCapabilities()* action on v2 *QosDevice* Services. The *ProtoTspec* parameter in the *QosDeviceExtendedState* state variable, that is returned as an output argument of *QD:GetExtendedQosState()* action of the 3.0 *QosDevice* Service, contains information about TSPECs that the *QosDevice* Service can support on a QoS Segment (e.g. *QosSegmentMaxDelayHigh*, *QosSegmentMaxJitter, QosSegmentMaxDelayLow)* [QOS DEVICE] .

The *QosManager* may use this information to decompose the end-to-end TSPEC parameters (e.g. *E2EMaxDelayHigh, E2EMaxJitter, E2EMaxDelayLow)* supplied by the Control Point into per-QoS Segment TSPEC parameters before invoking actions on individual *QosDevice* Services on the path to admit the stream. The *QosManager* examines if the sum of the QoS Segment specific parameters for all the QoS segments is less

than the end-to-end parameters to determine if the request for admission or update would be successful. Please refer to the [QoS Architecture] for more details on this.

For the *RequestedQosType* equal to "1" and "2", if *E2EMaxDelayHigh*, *E2EMaxDelayLow*, and, *E2EMaxJitter* parameters are supplied by the Control Point, the *QoS Manager* MAY provide the corresponding QoS segment specific parameters: *QosSegmentMaxDelayHigh*, *QosSegmentMaxDelayLow*, and, *QosSegmentMaxJitter*, respectively, in the TSPEC when invoking *QD:AdmitTrafficQos()* actions on *QosDevice* services on the path.

Some v2 or older UPnP *QosDevice* service instances may also expose the *QD:GetQosDeviceInfo()* action. The *QoS Manager* MAY invoke this action on the source and/or sink device to find out the port number and protocol information associated with that particular Traffic Descriptor. The *QD:GetExtendedQosState()* action on 3.0 *QosDevice* Services provide IP address, port number and protocol information associated with that particular Traffic Descriptor.

If the *QoS Manager* determines, based on the network capabilities that even the least preferred TSPEC of the *InitialTrafficDescriptor* cannot be admitted, the *QoS Manager* MUST return error to the Control Point with the Error Code 765.

If the *AdmitCtrlNet* parameter for an interface on a *QosDevice* Service is set to "1" in the *QosDeviceExtendedState* state variable, then the *QosManager* determines that the QoS Segment corresponding to that interface supports parameterized QoS.

If the *RequestedQosType* is set to "1" and the *QoS Manager* identifies that some of the QoS Segments on the path are not able to support parameterized QoS, the *QoS Manager* MUST establish prioritized QoS for the stream on prioritized QoS Segments by invoking either *QD:SetupTrafficQos()* or *QD:AdmitTrafficQos()* action for those QoS Segments and MUST establish parameterized QoS on parameterized QoS Segments by invoking *QD:AdmitTrafficQos()* action for those QoS Segments.

Note: If the *RequestedQosType* parameter is set to "1" and if the *QoS Manager* identifies that none of the QoS Segments on the path are able to support parameterized QoS, then the *QoS Manager* at the least establishes prioritized QoS for the stream on all the segments that support prioritized QoS.

If the *RequestedQosType* parameter is set to "2" and if the *QoS Manager* identifies that some of the QoS Segments on the path are not able to support parameterized QoS and no less preferred TSPEC is supplied, the *QoS Manager* MUST stop establishing QoS for the stream. The *QosManager* Service returns Error Code 782 to the Control Point in response to *RequestTrafficQos()* action as specified in section 2.4.1.2.2.

If the *QoS Manager* identifies that all of the *QosDevice* Services on the path are capable of Parameterized QoS, the *QoS Manager* MUST establish parameterized QoS on all of the QoS Segments by invoking *QD:AdmitTrafficQos()* action.

The *QoS Manager* MUST set the value of *ActiveTspecIndex* in the *TrafficDescriptor* to the index of the TSPEC for which it intends to established QoS before invoking *QD:AdmitTrafficQos()* and *QD:SetupTrafficQos()* actions.

The *QoS Manager* MUST establish QoS for the most preferred of the TSPECs that are admissible on all the *QosDevice* Services on the path from those supplied by the CP in the *InitialTrafficDescriptor.* The *QoS Manager* MUST NOT leave any unused allocated resources on the *QosDevice* Services.

If retries are performed for the same TSPEC, the *QoS Manager* MUST use random backoff.

If *E2EMaxDelayHigh* and *E2EMaxJitter* parameters are NOT supplied by the Control Point OR if the *RequestedQosType* equal to "1":

- If *QD:AdmitTrafficQos()* and/or *QD:SetupTrafficQos()* actions are successfully executed on all the *QosDevice* Services on the path, then the *UpdatedTrafficDescriptor* created by the *QosManager* MUST be returned to the Control Point as an output argument to the *RequestTrafficQos()* action. The *QoS*

Manager MUST set the value of *ActiveTspecIndex* in the *UpdatedTrafficDescriptor* to the index of the TSPEC for which the QoS is established.

If *E2EMaxDelayHigh* and *E2EMaxJitter* parameters are supplied by the Control Point for the *RequestedQosType* equal to "2", and if *QD:AdmitTrafficQos()* actions are successfully executed on all the *QosDevice* Services on the path, then:

- If the sum of all *MaxCommittedDelay* values and the sum of all *MaxCommittedJitter* values returned by *QosDevice* Services on the path is less than the *E2EMaxDelayHigh* and *E2EMaxJitter* values, respectively, then the *QosManager* MUST return the *UpdatedTrafficDescriptor* to the Control Point as an output argument to the *RequestTrafficQos()* action. The *QoS Manager* MUST set the value of *ActiveTspecIndex* in the *UpdatedTrafficDescriptor* to the index of the TSPEC for which the QoS is established.

- Otherwise, the *QoS Manager* MUST return error to the Control Point with Error Code 765.

If *QD:AdmitTrafficQos()* and/or *QD:SetupTrafficQos()* actions are unsuccessful on any of the *QosDevice* Services on the path, even after retries (if performed), then the *QoS Manager* MUST return error to the Control Point with Error Code 765.

As an output argument of *QD:AdmitTrafficQos()* action and the *QD:UpdateAdmittedTrafficQos()* action, a *QosDevice* Service may return a value in the *A_ARG_TYPE_Layer2Mapping* state variable [QOS DEVICE] This state variable is set by the *QosDevice* Service, when the *Layer2Mapping* information is available, to report the *Layer2StreamId* associated with the admitted stream on the QoS Segment. It is expected that at least one *QosDevice* Service on a QoS Segment knows about this mapping. If a *QosDevice* Service does not return a *Layer2StreamId* as output argument of *QD:AdmitTrafficQos()* action and the *QD:UpdateAdmittedTrafficQos()* action, then the *QoS Manager* MUST invoke *QD:SetL2Map()* action on that *QosDevice* Service if the action *QD:SetL2Map()* is implemented, and the *Layer2StreamId* mapping for that QoS Segment is known to the *QoS Manager*. This invocation of *QD:SetL2Map()*, when possible, serves to communicate the *Layer2StreamId* for this QoS Segment to the *QosDevice* Service.

2.4.1.4.5.1.QoS Admission Retry Mechanism (Informative)

If there is a failure on one of the *QD:AdmitTrafficQos()*actions (i.e. there is admission failure in any one of the QoS Segments), the *QoS Manager* may attempt to admit the stream by following the retry mechanism as outlined below.

- Before retrying to admit the stream with the same TSPEC or a less preferred TSPEC, the *QoS Manager* releases the network resources on QoS Segments where the stream was successfully admitted.

- The *QoS Manager* attempts to admit the stream with the supplied TSPECs in the preference order specified by the CP.

- The QoS Manager attempts to admit the stream with the same TSPEC that it tried earlier one more time. If the *QoS Manager* retries to admit the stream with the TSPEC that it used in its earlier attempt, the *QoS Manager* waits for a period of time determined by a random backoff algorithm after the clean up before retrying the subsequent admission.

- If the *QoS Manager* retries to admit the stream with a less preferred TSPEC than what it used in its earlier attempt, the *QoS Manager* waits for a period of time determined by a random backoff algorithm after the clean up on the entire path before retrying the subsequent admission.

- Once the *QoS Manager* successfully admits the stream with one of the supplied Tspecs, the *QoS Manager* stops retrying.

- If the *QoS Manager* identifies that the stream cannot be admitted within 30 secs, the *QoS Manager* needs to release the resources on QoS Segments where the stream was successfully admitted by calling

QD:ReleaseAdmittedTrafficQos() or *QD:ReleaseTrafficQos()* action on 3.0 *QosDevice* Services & *QD:ReleaseTrafficQos()* on 2.0 or older *QosDevice* Services.

Suggested random backoff algorithm is as follows:

The duration of each backoff interval is a random number in the interval [50, 100*2^n] milliseconds, where n = retry attempt and take values of {1, 2}. Thus, there will be 2 retry attempts after initial failure. Thus, the maximum backoff time will be 200+400= 600 milliseconds, after which the *QoS Manager* gives up setting up of QoS. These numbers should be configurable in an out-of-band manner.

When the *QoS Manager* is performing preemption, which is explained later in Section 2.4.6.4.3.1 below, n takes values of {1, 2, 3}. Thus there will be 3 preemption attempts before the *QoS Manager* reports an error to the Control Point.

2.4.1.5. Dependency on State (if any)

Since UPnP *QosManager* is defined as a stateless entity, there is no dependency of this action on state.

2.4.1.6. Effect on State (if any)

Since UPnP *QosManager* is defined as a stateless entity, there is no effect of this action on state

2.4.1.7. Errors

Table 2-11: Error Codes for *RequestTrafficQos()*

errorCode	errorDescription	Description
701	TrafficHandle must not be specified by the Control Point	TrafficHandle must not be specified by the Control Point
710	Incomplete TrafficId	At least one of the required parameters in the TrafficId is absent.
711	TrafficId could not be completed	The QosManager could not complete the TrafficId
713	Malformed Name-string	One or more of the String fields in the traffic descriptor do not conform to the specification.
715	Policy parameters MUST NOT be specified by the Control Point	PolicyLastModified, PolicyModifyingUserName, or PolicyHolderConfigUrl MUST NOT be specified by the Control Point
716	An input parameter (e.g. TrafficDescriptor) does not validate against the XML schema	One of the XML-based input arguments does not follow the schema
718	Invalid Requested QoS Type	Value supplied for RequestedQosType parameter is invalid

errorCode	errorDescription	Description
721	TrafficImportanceNumber MUST not be specified by the Control Point	TrafficImportanceNumber must not be specified by the Control Point
722	The ActiveTspecIndex MUST NOT be specified by the Control Point	The ActiveTspecIndex must not be specified by the Control Point
725	Duplicate TspecIndex	All Tspec indices must be unique
740	No QosDevice at Source IP address	Source QosDevice does not exist.
741	No QosDevice at Destination IP address	Destination QosDevice does not exist.
742	No QosDevice at QosBoundarySourceAddress	Source Boundary QosDevice does not exist.
743	No QosDevice at QosBoundaryDestinationAddress	Destination Boundary QosDevice does not exist.
744	No QosDevices available	No QosDevice instances found on the network.
746	QosPolicyHolder failure	QosManager gets incorrect information from the QosPolicyHolder (could be malformed XML)
752	QosBoundarySourceAddress or QosBoundaryDestinationAddress not on the same subnet.	The source and/or destination is not on the subnet which is under the management of this QosManager
765	Admission Failure	A stream cannot be admitted due to lack of resources on the network.
772	Network not capable	Request exceeds network capabilities
773	Cannot retrieve XML namespace	An XML namespace was included in one of the arguments, but the QosManager service was unable to retrieve the namespace from the location provided.
780	QosPolicyHolder not found	The QosPolicyHolder specified by the PolicyHolderId was not found.
781	Preferred QosPolicyHolder not found	The Preferred QosPolicyHolder not found on the network.
782	Parameterized QoS not possible	End to end parameterized QoS cannot be established for a stream due to presence of at least on prioritized QoS segment.
785	Different Preferred QosPolicyHolder ServiceId values	QosDevice Services report different ServiceId values for the Preferred QosPolicyHolder with the same highest preference count.

errorCode	errorDescription	Description
791	TrafficLeaseTime is missing	TrafficLeaseTime in the traffic descriptor is missing
792	Incorrect TSPEC	Input parameters supplied in the TSPEC are incorrect.

2.4.2. UpdateTrafficQos()

When a Control Point needs to change the QoS associated with a particular traffic, it uses the *UpdateTrafficQos()* action.

Upon receiving *UpdateTrafficQos()* action, the *QosManager* repeats the admission control process described above for the revised TrafficDescriptor.

Note: It is possible that Control Points can invoke *UpdateTrafficQos()* action on a *QosManager:2* Service to update traffic descriptors with *RequestedQosType* greater than "0". However, the behavior of the *QosManager:2* Service upon invocation of such action is unspecified. Hence, invocation of *UpdateTrafficQos()*action by Control Points on *QosManager:2* Service for traffic descriptors with *RequestedQosType* greater than "0" is discouraged.

Table 2-12 shows which elements may be updated by the Control Point when calling the *UpdateTrafficQos()* action.

Table 2-12 Elements that are allowed to be updated when calling UpdateTrafficQos()

Parameter Name	May be Updated?	Comments
TrafficHandle	No	
TrafficId	No	Not applicable / See sub–elements below
. SourceAddress	No	Must be the same one as from the returned Traffic Descriptor
. SourcePort	No	Must be the same one as from the returned Traffic Descriptor
. DestinationAddress	No	Must be the same one as from the returned Traffic Descriptor
. DestinationPort	No	Must be the same one as from the returned Traffic Descriptor
. IpProtocol	No	Must be the same one as from the returned Traffic Descriptor
AvailableOrderedTspecList	Yes	See sub-elements below
. Tspec	Yes	See sub-elements below
. . TspecIndex	Yes	
. . AvTransportUri	No	
. . AvTransportInstanceId	No	
. . TrafficClass	No	
. . v2TrafficSpecification	No	Not applicable/ See sub-elements below
. . . v3TrafficSpecification	No	Not applicable/ See sub-elements below
. . . . RequestedQosType	Yes	

Parameter Name	May be Updated?	Comments
. . . . DataRate	Yes	
. . . . MaxBurstSize	Yes	
. . . . MinServiceRate	Yes	
. . . . ReservedServiceRate	Yes	
. . . . TimeUnit	Yes	
. . . . MaxPacketSize	Yes	
. . . . E2EMaxDelayHigh	Yes	
. . . . E2EMaxDelayLow	Yes	
. . . . E2EMaxJitter	Yes	
. . . . QosSegmentSpecificParameters	N/A	Not applicable
. . . . . InterfaceId	N/A	Must not be supplied by Control Point
. . . . . QosSegmentId	N/A	Must not be supplied by Control Point
. . . . . QosSegmentMaxDelayHigh	N/A	Must not be supplied by Control Point
. . . . . QosSegmentMaxDelayLow	N/A	Must not be supplied by Control Point
. . . . . QosSegmentMaxJitter	N/A	Must not be supplied by Control Point
. . . . MaxServiceInterval	Yes	
. . . . MinServiceInterval	Yes	
. . . . LossSensitivity	Yes	
. . . . ServiceType	Yes	
ActiveTspecIndex	No	QosManager may update this in its response
TrafficImportanceNumber	No	QosManager may update this in its response
QosBoundarySourceAddress	No	
QosBoundaryDestinationAddress	No	
MediaServerConnectionId	No	
MediaRendererConnectionId	No	
TrafficLeaseTime	Yes	
PolicyHolderId	Yes	
PolicyLastModified	N/A	Must not be supplied by Control Point
PolicyModifyingUserName	N/A	Must not be supplied by Control Point
PolicyHolderConfigUrl	N/A	Must not be supplied by Control Point
QosBoundarySourceUuid	No	
QosBoundaryDestinationUuid	No	
Critical	Yes	

Parameter Name	May be Updated?	Comments
PolicyHolderConsultedId	No	QosManager may update this in its response
PolicyHolderConsultedType	No	QosManager may update this in its response
OptionalPolicyParams	Not applicable	See sub-elements below
. UserName	Yes	
. VendorApplicationName	No	
. PortName	No	
. ServiceProviderServiceName	No	
. CpName	Yes	This MUST be the CpName of the Control Point invoking the update

2.4.2.1. Arguments

Table 2-13: Arguments for UpdateTrafficQos()

Argument	Direction	relatedStateVariable
TrafficHandle	In	A_ARG_TYPE_TrafficHandle
RequestedTrafficDescriptor	In	A_ARG_TYPE_TrafficDescriptor
ImplementedTrafficDescriptor	Out	A_ARG_TYPE_TrafficDescriptor
NumPolicyHolders	Out	A_ARG_TYPE_NumPolicyHolders

2.4.2.2. Service Requirements

2.4.2.2.1. Base Requirements

The following generic requirements apply irrespective of the absence or presence and value of *RequestedQosType* parameter in the *RequestedTrafficDescriptor* input argument:

- If Control Point supplies *TrafficImportanceNumber* in TrafficDescriptor to the *QosManager* Service when calling the *UpdateTrafficQos()* action, *QosManager* returns error 721.
- If Control Point supplies *ActiveTspecIndex* in TrafficDescriptor to *QosManager* when calling the *UpdateTrafficQos()* actions, the *QosManager* Service MUST return error code 722.
- The *QosManager* Service MUST include a valid *ActiveTspecIndex* value in the TrafficDescriptor when it returns a non error value in response to *UpdateTrafficQos()* action.
- If a Control Point attempts to update elements in the TrafficDescriptor that are not specified by Table 2-12, the *QosManager* Service MUST return error code 714.
- In response to the Control Point calling the *UpdateTrafficQos()* action, the *QosManager* may update *ActiveTspecIndex*, and/or *TrafficImportanceNumber* in the TrafficDescriptor structure. All other non-<any> XML elements MUST NOT be updated by the *QosManager*.
- If a Control Point supplies a non-existent *TrafficHandle* as first argument of *UpdateTrafficQos()*, the *QosManager* MUST return an error (Error Code 703).

- If a Control Point supplies *PolicyLastModified*, *PolicyModifyingUserName*, or *PolicyHolderConfigUrl* when calling the *UpdateTrafficQos()* action, the *QosManager* MUST return error code 715.

- If a Control Point does not supply a unique *TspecIndex* for every TSPEC within an *AvailableOrderedTspecList*, then the *QosManager* will return the error 725.

- The *QosManager* MUST include the TrafficPolicy parameters received from the *QPH:GetTrafficPolicy()* in the TrafficDescriptor when successfully returning the *UpdatedTrafficDescriptor*. The *QosManager* MUST always include *QosPolicyHolderId*.

- If the *QoS Manager* is successful in updating the traffic stream on all the 3.0 *QosDevice* Services on the path, the *QosManager* Service MUST return the updated TrafficDescriptor to the Control Point as *ImplementedTrafficDescriptor* output argument of *QM:UpdateTrafficQos()* action.

- The *QosManager* Service MUST populate all the fields of the *TrafficId* in the *ImplementedTrafficDescriptor*.

- If the *QoS Manager* obtains the *MaxCommittedDelay* for every QoS Segment on the path, then the *QosManager* Service MUST populate *E2EMaxDelayHigh* in the *ImplementedTrafficDescriptor*.

- If the *QoS Manager* obtains the *MaxCommittedJitter* for every QoS Segment on the path, then the *QosManager* Service MUST populate the *E2EMaxJitter* in the *ImplementedTrafficDescriptor*.

- The *QosManager* Service MUST NOT return *QosSegmentSpecificParameter* in the *ImplementedTrafficDescriptor.*

- The *QosManager* Service MUST return the value of "1" for the *NumPolicyHolders* output argument if it uses the preferred *QosPolicyHolder* Service OR the *QosPolicyHolder* Service supplied by the Control Point OR a single available *QosPolicyHolder* Service on the network. Otherwise, (i.e, if the *QoS Manager* uses default policies) the *QosManager* Service MUST return the actual number of *QosPolicyHolder* Services discovered on the network. The actual *QosPolicyHolder* Service and its type used by the *QosManager* Service are identified by the *PolicyHolderConsultedId* and *PolicyHolderConsultedType* parameters in the UpdatedTrafficDescriptor output argument.

- If *UpdateTrafficLeaseTime* in the TrafficDescriptor is set to **"1"**, then the *QoS Manager* MUST invoke *QD:UpdateTrafficLeaseTime()* on each *QosDevice* Service on the path and MUST NOT invoke *QD:UpdateAdmittedQos()*

2.4.2.2.2. *RequestedQosType Specific Requirements*

The following requirements apply when *RequestedQosType* parameter is present in the *RequestedTrafficDescriptor* input parameter:

- For the *RequestedQosType* parameter with the value of "1" or "2" in the *RequestedTrafficDescriptor*, if the *QoS Manager* is unable to successfully update the traffic stream on all the 3.0 *QosDevice* Services on the path, even after retries (if performed), the *QosManager* Service MUST return Error Code 763 to the Control Point.

- For the *RequestedQosType* parameter with the value of "2" in the *RequestedTrafficDescriptor*, if the *QoS Manager* identifies that some of the QoS Segments on the path are not able to support Parameterized QoS, if no less preferred TSPEC is supplied, the *QosManager* Service MUST return error to the CP in response to *UpdateTrafficQos()* action with Error Code 782.

- If a Control Point supplies the *RequestedQosType* parameter with the value greater than "2" in the *RequestedTrafficDescriptor*, the *QosManager* Service MUST return error with Error Code 718 to the Control Point.

- For the *RequestedQosType* parameter with the value of "1" or "2", if a Control Point does not supply a value for the *TrafficLeaseTime* field in the *RequestedTrafficDescriptor* input argument, the *QosManager* Service MUST return error with Error Code 791 to the Control Point.

- For the *RequestedQosType* parameter with the value of "1" or "2", if a Control Point supplies the value for the *E2EMaxDelayLow* parameter but does NOT supply the value for the *E2EMaxDelayHigh* in the input TSPEC parameters, the *QosManager* Service MUST return error with Error Code 792 to the Control Point.

2.4.2.3. Control Point requirements when calling the action

2.4.2.3.1. Base Requirements

The following generic requirements apply irrespective of the absence or presence and value of *RequestedQosType* parameter in the *RequestedTrafficDescriptor* input parameter:

- A Control Point's updated TrafficDescriptor MUST be conformant to Table 2-12. The fields that cannot be updated MUST have the same values as contained in the traffic descriptor stored on the *QosDevice* Service that is identified by the *TrafficHandle*.

- A Control Point MUST supply a valid *TrafficHandle* as the first argument when calling the *UpdateTrafficQos()* action.

- A Control Point MUST NOT supply *TrafficImportanceNumber* in the TrafficDescriptor to *QosManager* when calling the *UpdateTrafficQos()* action.

- A Control Point MUST supply a unique *TspecIndex* for every TSPEC within an *AvailableOrderedTspecList*.

- A Control Point MUST NOT supply *ActiveTspecIndex* in TrafficDescriptor to *QosManager* when calling the *UpdateTrafficQos()* actions.

- Whenever a Control Point specifies a *PolicyHolderId* it MUST specify a valid Id for a PolicyHolder as defined in the *QosPolicyHolder* service definition.[POLICY HOLDER]

- A Control Point MUST NOT supply *PolicyLastModified*, *PolicyModifyingUserName*, *PolicyHolderConfigUrl*.

- If a Control Point wants to update the *TrafficLeaseTime* of the QoS reservation for a stream indicated by the TrafficDescriptor without making any other updates to the QoS reservation, the *UpdateTrafficLeaseTime* field in the TrafficDescriptor MUST be set to **"1"**.

2.4.2.3.2. RequestedQosType Specific Requirements

The following requirements apply when *RequestedQosType* parameter is present in the *RequestedTrafficDescriptor* input parameter:

- A Control Point MUST NOT supply a value greater than "2" for the *RequestedQosType parameter.*

- For the *RequestedQosType* parameter with the value of "1" or "2", a Control Point MUST supply a value for the *TrafficLeaseTime* field in the *RequestedTrafficDescriptor* input argument.

- For the *RequestedQosType* parameter with the value of "1" or "2", if a Control Point supplies a value for the *E2EMaxDelayLow* as one of the input TSPEC parameters, then the Control Point MUST also supply the value for the *E2EMaxDelayHigh* parameters in the TSPEC.

2.4.2.4. QoS Manager Requirements

2.4.2.4.1. Base requirements

The following generic requirements apply irrespective of the absence or presence and value of *RequestedQosType* parameter in the *RequestedTrafficDescriptor* input parameter:

- When a Control Point calls the *UpdateTrafficQos()* action and the *QoS Manager* subsequently invokes actions with a TrafficDescriptor as an argument, the *QoS Manager* MUST NOT modify those elements that MUST equal the original TrafficDescriptor according to Table 2-9.
- In addition, the *QosManager* MUST use the original TrafficHandle and MUST NOT generate a new TrafficHandle.

2.4.2.4.2. Prioritized QoS Update Requirements

Upon receiving *UpdateTrafficQos()* action with the *RequestedQosType* parameter in the *RequestedTrafficDescriptor* either absent or with a value of 0, the *QoS Manager* MUST perform the following steps:

- The *QoS Manager* MUST obtain QoS traffic policies as specified in Section 2.4.1.4.2.
- The *QoS Manager* then MUST invoke *QD:ReleaseTrafficQos()* and *QD:SetupTrafficQos()* actions, subsequenctly, on all v2 *QosDevice* Services on the path. The *QoS Manager* MUST invoke *QD:UpdateAdmittedQos()* OR *QD:ReleaseTrafficQos() followed by QD:SetupTrafficQos(),* action on all v3 *QosDevice* Services on the path. It is recommended that the *QoS Manager* invokes *QD:UpdateAdmittedQos() action on v3 QosDevice Services.*
- If *QD:UpdateAdmittedQos()* and *QD:SetupTrafficQos()* actions are successfully executed on all the *QosDevice* Services on the path, then the updated TrafficDescriptor created by the *QosManager* is returned to the Control Point as *ImplementedTrafficDescriptor* output argument of *QM:UpdateTrafficQos()* action. The *QoS Manager* MUST set the value of ActiveTspecIndex in the *ImplementedTrafficDescriptor* to the index of the TSPEC for which the QoS is established.

2.4.2.4.3. Parameterized & Hybrid QoS Update Requirements

Upon receiving *UpdateTrafficQos* action with the *RequestedQosType* parameter in the *RequestedTrafficDescriptor* of either "1" or "2", the *QoS Manager* MUST perform the following steps:

- If the *RequestedQosType* parameter is "1", the *QoS Manager* MUST obtain QoS traffic policies as specified in Section 2.4.1.4.3.
- If the *RequestedQosType* parameter is "2", the *QoS Manager* MAY obtain QoS traffic policies as specified in Section 2.4.1.4.3.
- If the *RequestedQosType* parameter is "1", the *QoS Manager* MUST invoke *QD:ReleaseTrafficQos()* and *QD:SetupTrafficQos()* subsequenctly on all v2 *QosDevice* Services on the path and the *QoS Manager* MUST invoke *QD:UpdateAdmittedQos()* action on all v3 *QosDevice* Services on the path.
- If the *RequestedQosType* parameter is "2", the *QoS Manager* MUST invoke *QD:UpdateAdmittedQos()* action on all v3 *QosDevice* Services on the path.
- For the *RequestedQosType* equal to "1" and "2", if *E2EMaxDelayHigh*, *E2EMaxDelayLow*, and, *E2EMaxJitter* parameters are supplied by the Control Point, the *QoS Manager* MUST provide the corresponding QoS segment specific parameters: *QosSegmentMaxDelayHigh*,

QosSegmentMaxDelayLow, and, *QosSegmentMaxJitter*, respectively, in the TSPEC when invoking *QD:UpdateAdmittedQos()* actions on *QosDevice* services on the path.

- The *QoS Manager* MUST establish QoS for the most preferred of the TSPECs that are admissible on all the *QosDevice* Services on the path from those supplied by the Control Point in the *RequestedTrafficDescriptor*. The *QoS Manager* MUST NOT leave any unused allocated resources on the *QosDevice* Services.

- If *E2EMaxDelayHigh* and *E2EMaxJitter* parameters are NOT supplied by the Control Point OR if the *RequestedQosType* equal to "0" or "1:

    - If *QD:UpdateAdmittedQos()* and *QD:SetupTrafficQos()* actions are successfully executed on all the *QosDevice* services on the path, then the updated TrafficDescriptor created by the *QosManager* is returned to the Control Point as *ImplementedTrafficDescriptor* output argument of *QM:UpdateTrafficQos()* action. The *QoS Manager* MUST set the value of *ActiveTspecIndex* in the *ImplementedTrafficDescriptor* to the index of the TSPEC for which the QoS is established.

- If *E2EMaxDelayHigh* and *E2EMaxJitter* parameters are supplied by the Control Point for the *RequestedQosType* equal to "2", and if the *QD:AdmitTrafficQos()* and/or *QD:SetupTrafficQos()* actions are successfully executed on all the *QosDevice* Services on the path, then:

    - If the sum of all *MaxCommittedDelay* values and the sum of all *MaxCommittedJitter* values returned by *QosDevice* Services on the path is less than the *E2EMaxDelayHigh* and *E2EMaxJitter* values, respectively, then the *QosManager* MUST return the *ImplementedTrafficDescriptor* to the Control Point as an output argument of the *UpdateTrafficQos()* action. The *QoS Manager* MUST set the value of *ActiveTspecIndex* in the *UpdatedTrafficDescriptor* to the index of the TSPEC for which the QoS is established.

    - Otherwise, the *QoS Manager* MUST return Error code 763 to the Control Point in response to the *UpdatedTrafficQos()* action, even after retries (if performed), and the *QoS Manager* MUST restore the QoS established for the stream prior to attempting the update on all *QosDevice* Services on the path.

- As an output argument of the *QD:UpdateAdmittedTrafficQos()* action, a *QosDevice* Service may return a value in the *A_ARG_TYPE_Layer2Mapping* state variable [QOS DEVICE] . This state variable is set by the *QosDevice* Service, when the *Layer2Mapping* information is available, to report the *Layer2StreamId* associated with the admitted stream on the QoS Segment. It is expected that at least one *QosDevice* Service on a QoS Segment knows about this mapping. If a *QosDevice* Service does not return a Layer2StreamId as output argument of the *QD:UpdateAdmittedTrafficQos()* action, then the *QoS Manager* MUST invoke *QD:SetL2Map()* action on that *QosDevice* Service if the action *QD:SetL2Map()* is implemented, and the *Layer2StreamId* mapping for that QoS Segment is known to the *QosManager*. This invocation of *QD:SetL2Map()*, when possible, serves to communicate the *Layer2StreamId* for this QoS Segment to the *QosDevice* Services on this QoS Segment.

- If there is a failure on one of the *QD:UpdateAdmittedQos()* actions (i.e. there is admission failure in any one of the QoS Segments), the *QoS Manager* MAY follow the steps as identified in Section 2.4.1.4.5.1 to retry admitting the updated traffic descriptor.

- If retries are performed for the same TSPEC, the *QoS Manager* MUST use random backoff.

- If the *QoS Manager* determines that the stream cannot be successfully updated even with the least preferred TSPEC, the *QosManager* MUST return Error code 763 to the Control Point in response to the *UpdatedTrafficQos()* action and the *QoS Manager* MUST restore the QoS established for the stream prior to attempting the update on all *QosDevice* Services on the path.

2.4.2.5. Dependency on State (if any)

Since UPnP *QosManager* is defined as a stateless entity, there is no dependency for this action on the state.

2.4.2.6. Effect on State (if any)

Since *QosManager* is defined as a stateless entity, there is no effect of this action on state.

2.4.2.7. Errors

Table 2-14: Error Codes for UpdateTrafficQos

errorCode	errorDescription	Description
703	Traffic Handle unknown to this device	Traffic Handle unknown to this device
713	Malformed Name-string	One or more of the String fields in the traffic descriptor do not conform to the specification.
714	Tried to update unmodifiable Traffic Descriptor elements	QosManager cannot be requested to update some of the TrafficDescriptor parameters e.g. TrafficId, TrafficHandle
715	Policy parameters must not be specified by the Control Point	PolicyLastModified, PolicyModifyingUserName, or PolicyHolderConfigUrl MUST NOT be specified by the Control Point
716	An input parameter (e.g. TrafficDescriptor) does not validate against the XML schema	One of the XML-based input arguments does not follow the schema
718	Invalid Requested QoS Type	Value supplied for RequestedQosType parameter is invalid
721	TrafficImportanceNumber must not be specified by the Control Point	TrafficImportanceNumber must not be specified by the Control Point
722	The ActiveTspecIndex must not be specified by the Control Point	The ActiveTspecIndex must not be specified by the Control Point
725	Duplicate TspecIndex	All Tspec indices must be unique
740	No QosDevice at Source IP address	Source QosDevice does not exist.
741	No QosDevice at Destination IP address	Destination QosDevice does not exist.
742	No QosDevice at QosBoundarySourceAddress	Source Boundary QosDevice does not exist.
743	No QosDevice at QosBoundaryDestinationAddress	Destination Boundary QosDevice does not exist.
744	No QosDevices available	No QosDevice instances found on the network.
746	QosPolicyHolder failure	QosManager gets incorrect information from the QosPolicyHolder (could be malformed XML)
752	QosBoundarySourceAddress or QosBoundaryDestinationAddress not on the same subnet	The source and/or destination is not on the subnet which is under the management of this QosManager

errorCode	errorDescription	Description
763	Update Failed	Failure in updating the traffic stream on one or more QosDevice Service(s) on the path.
773	Cannot retrieve XML namespace	An XML namespace was included in one of the arguments, but the service was unable to retrieve the namespace from the location provided.
780	QosPolicyHolder not found	The QosPolicyHolder specified by the PolicyHolderId was not found.
781	Preferred QosPolicyHolder not found	The Preferred QosPolicyHolder was not found on the network.
782	Parameterized QoS not possible	End to end parameterized QoS cannot be established for a stream due to presence of at least on prioritized QoS segment.
785	Different Preferred QosPolicyHolder ServiceId values	QosDevice Services report different ServiceId values for the Preferred QosPolicyHolder with the same highest preference count.
791	TrafficLeaseTime is missing	TrafficLeaseTime in the traffic descriptor is missing
792	Incorrect TSPEC	Input parameters supplied in the TSPEC are incorrect.

2.4.3. ReleaseTrafficQos()

Control Point can invoke this action for releasing the Quality of Service for a particular traffic stream.

2.4.3.1. Arguments

Table 2-15: Arguments for ReleaseTrafficQos

Argument	Direction	relatedStateVariable
RevokeTrafficHandle	In	A_ARG_TYPE_TrafficHandle

2.4.3.2. Service requirements

If a Control Point supplies an unknown TrafficHandle, the *QosManager* returns an error (Error Code 703)

2.4.3.3. Control Point requirements when calling the action

A Control Point MUST supply a valid Traffic Handle when calling *ReleaseTrafficQos()*

2.4.3.4. QoS Manager requirements

Upon receiving *ReleaseTrafficQos()* for a traffic stream identified by the *RevokeTrafficHandle* argument, the *QoS Manager* MUST invoke the *QD:ReleaseTrafficQos()* action on all v2 *QosDevice* services and *QD:ReleaseAdmittedQos()* or *QD:ReleaseTrafficQos()* action on all v3 *QosDevice* services on the path of the traffic stream. It is recommended that the *QoS Manager* invokes *QD:ReleaseAdmittedQos()* on v3 *QosDevice* services.

2.4.3.5. Dependency on State (if any)

Since *QosManager* is defined as a stateless entity, there is no dependency for this action on the state.

2.4.3.6. Effect on State (if any)

Since *QosManager* is defined as a stateless entity, there is no effect of this action on state.

2.4.3.7. Errors

Table 2-16: Error Codes for ReleaseTrafficQos

errorCode	errorDescription	Description
703	Traffic Handle unknown to this device	
744	No QosDevice services available	
773	Cannot retrieve XML namespace	An XML namespace was included in one of the arguments, but the service was unable to retrieve the namespace from the location provided.

2.4.4. BrowseAllTrafficDescriptors()

A Control Point can invoke this action to browse all the 'TrafficDescriptors' configured on the network. Each TrafficDescriptor represents QoS for a particular traffic stream.

2.4.4.1. Arguments

Table 2-17: Arguments for BrowseAllTrafficDescriptors

Argument	Direction	relatedStateVariable
NumberOfTrafficDescriptors	Out	A_ARG_TYPE_NumTrafficDescriptors
TrafficDescriptorList	Out	A_ARG_TYPE_ ListOfTrafficDescriptors

2.4.4.2. QoS Manager requirements

When a Control Point calls the *BrowseAllTrafficDescriptors* action, the *QoS Manager* MUST report back the details on various traffic streams admitted on the *QosDevice* services on the network by querying known *QosDevice* services (for example by calling the action *QD:GetQosState()* or *QD:GetExtendedQosState()*).

2.4.4.3. Dependency on State (if any)

This action does not have any dependency on state.

2.4.4.4. Effect on State (if any)

This action does not have any effect on state.

2.4.4.5. Errors

Table 2-18: Error Codes for BrowseAllTrafficDescriptors

errorCode	errorDescription	Description
704	Network contains Different TrafficDescriptors for same TrafficHandle	Network contains Different TrafficDescriptors for same TrafficHandle. **Note:** This Error Code is maintained forward for backwards compatibility.
773	Cannot retrieve XML namespace	An XML namespace was included in one of the arguments, but the service was unable to retrieve the namespace from the location provided.

2.4.5. GetQmCapabilities()

A Control Point can invoke this action to determine various optional capabilities supported by the *QosManager* Service (e.g. preemption).

2.4.5.1. Arguments

Table 2-19: Arguments for GetQmCapabilities

Argument	Direction	relatedStateVariable
QmCapabilities	Out	A_ARG_TYPE_QmCapabilities

2.4.5.2. QosManager Service requirements

If the *QosManager* Service is capable of reporting information about Blocking Streams while attempting to admit a traffic stream as specified in Section 2.4.6.4.3.1 below, the *QosManager* Service MUST return the value of *ReportBlockingStreams* field as "1" in the *QmCapabilities* state variable that is returned as an output argument of *GetQmCapabilities()* action. Otherwise the *QosManager* Service MUST return the value of *ReportBlockingStreams* field as "0".

If the *QosManager* Service is capable of performing Preemption of Blocking Streams while attempting to admit a traffic stream as specified in Section 2.4.6.4.3.1 below, the *QosManager* Service MUST return the value of *Preemption* field as "1" in the *QmCapabilities* state variable that is returmed as an output argument of *GetQmCapabilities()* action. Otherwise the *QosManager* Service MUST return the value of *Preemption* field as "0".

2.4.5.3. Dependency on State (if any)

This action does not have any dependency on state.

2.4.5.4. Effect on State (if any)

This action does not have any effect on state.

2.4.5.5. Errors

Table 2-20: Error Codes for GetQmCapabilities

errorCode	errorDescription	Description

2.4.6. RequestExtendedTrafficQos()

A Control Point invokes this optional action for setting up QoS for a specified traffic stream with optional additional capabilities, such as preemption and reporting of blocking streams that are not supported by *RequestTrafficQos()* action.

2.4.6.1. Arguments

Table 2-21: Arguments for RequestExtendedTrafficQos

Argument	Direction	relatedStateVariable
InitialTrafficDescriptor	In	A_ARG_TYPE_TrafficDescriptor
SelectedQmCapabilities	In	A_ARG_TYPE_QmCapabilities
TrafficHandle	Out	A_ARG_TYPE_TrafficHandle
UpdatedTrafficDescriptor	Out	A_ARG_TYPE_TrafficDescriptor
ExtendedTrafficQosInfo	Out	A_ARG_TYPE_ExtendedTrafficQosInfo
ResultedQosType	Out	A_ARG_TYPE_ResultedQosType

2.4.6.2. Service requirements

2.4.6.2.1. Base Requirements

Refer to Section 2.4.1.2.1 above for base requirements when a Control Point invokes *RequestExtendedTrafficQos()* action.

Note: The *QosManager* Service does not return *NumPolicyHolders* parameter as an output argument of *RequestExtendedTrafficQos()* action.

2.4.6.2.2. Additional Requirements

If a Control Point requests non-supported *QosManager* Service capability in the SelectedQmCapabilities input argument, the *QosManager* Service MUST return Error Code 790.

For *RequestedQosType* parameter with the value of "1", if the *QoS Manager* identifies that one of the QoS Segments on the path is not capable of supporting Parameterized QoS, the *QosManager* Service MUST set *ResultedQosType* output argument to "1".

Note: The *QoS Manager* determines whether a QoS Segment is capable of supporting parameterized QoS based on the *AdmitCntrlNet* parameter retrieved via the QD:*GetExtendedQosState()* action.

For *RequestedQosType* parameter with the value of "1", if the *QoS Manager* identifies that all of QoS Segments on the path are capable of supporting Parameterized QoS, the *QosManager* Service MUST set *ResultedQosType* output argument to "2".

For the *RequestedQosType* parameter with the value of "2", if the *QoS Manager* identifies that some of the QoS Segments on the path are not able to support Parameterized QoS, and no less preferred TSPEC is supplied, the *QosManager* Service MUST return error to the Control Point in response to *RequestExtendedTrafficQos()* with error code of 782.

If a Control Point supplies the *RequestedQosType* parameter with the value greater than "2" in the *RequestedTrafficDescriptor*, the *QosManager* Service MUST return error with Error Code 718 to the Control Point.

For the *RequestedQosType* parameter with the value of "1" or "2", if a Control Point does not supply a value for the *TrafficLeaseTime* field in the *InitialTrafficDescriptor* input argument, the *QosManager* Service MUST return error with Error Code 791 to the Control Point.

2.4.6.3. Control Point requirements when calling the action

2.4.6.3.1. Base Requirements

Refer to Section 2.4.1.3.1 above when invoking *RequestExtendedTrafficQos()* action.

2.4.6.3.2. Additional Requirements

For the *RequestedQosType* parameter with the value of "1" or "2", a Control Point MUST supply a value for the *TrafficLeaseTime* field in the *InitialTrafficDescriptor* input argument.

For the *RequestedQosType* parameter with the value of "1" or "2", if a Control Point supplies a value for the *E2EMaxDelayLow* as one of the input TSPEC parameters, then the Control Point MUST also supply the value for the *E2EMaxDelayHigh* parameters in the TSPEC.

2.4.6.4. QoS Manager requirements

2.4.6.4.1. Base requirements

When a Control Point calls the *RequestExtendedTrafficQos()* action, the *QoS Manager* MUST follow all the requirements identified in Section 2.4.1.4.1 above, irrespective of the value of *RequestedQosType* parameter in the *InitialTrafficDescriptor* input parameter.

2.4.6.4.2. Prioritized QoS Establishment Requirements with Extended Functionality

Upon receiving the *RequestExtendedTrafficQos()* action with the *RequestedQosType* parameter in the *InitialTrafficDescriptor* of value "0", the *QoS Manager* MUST follow the requirements identified in Section 2.4.1.4.4 above to establish prioritized QoS for the stream identified by *InitialTrafficDescriptor* input parameter.

2.4.6.4.3. Parameterized and Hybrid QoS Establishment Requirements with Extended Functionality

Upon receiving the *RequestExtendedTrafficQos()* action with the *RequestedQosType* parameter in the *InitialTrafficDescriptor* of value either "1" or "2", the *QoS Manager* MUST follow the requirements identified in Section 2.4.1.4.5 to admit the stream identified by the *InitialTrafficDescriptor*.

However, if the *QoS Manager* determines that the traffic stream cannot be admitted even with the least preferred TSPEC and if the CP requested extended *QoS Manager* capabilities (i.e. either Preemption =1 or ReportBlockingStreams =1 in the SelectedQmCapabilities input argument), the *QoS Manager* MUST perform the extended functions as specified in Section 2.4.6.4.3.1 to admit the stream with the least preferred TSPEC. If

the CP didn't request extended *QoS Manager* capabilities (i.e. CP set Preemption = "0" and ReportBlockingStreams = "0"), the *QoS Manager* MUST NOT perform the extended functions.

2.4.6.4.3.1. Additional QoS Manager Capabilities Functions

2.4.6.4.3.1.1.Implementation Guidelines (Informative):

- Upon failure of *QD:AdmitTrafficQos()* or *QD:UpdateAdmittedQos()* action, the *QosDevice* Service provides Layer2StreamIds of Blocking Traffic Streams, if available, in the ListOfLayer2StreamIds field of *A_ARG_TYPE_AdmitTrafficQosExtendedResult* output argument to the *QoS Manager*. See [QOS DEVICE] for more details.

- The *QoS Manager* invokes *QD:GetExtendedQosState()* on *QosDevice* Services on the Blocked QoS Segment(s) to obtain UPnP-QoS TrafficDescriptors corresponding to the blocking Layer2StreamIds.

- The *QoS Manager* invokes *QPH:GetListOfTrafficPolicies()* action to obtain the *UserImportanceNumber* for each of the blocking TrafficDescriptors and the *InitialTrafficDescriptor*; only if the version of the *QosPolicyHolder* Service consulted in section 2.4.1.4.3 is 3.0 or higher.

2.4.6.4.3.1.2.Requirements for Reporting Blocking Streams:

If a Control Point sets the ReportBlockingStream field to a value of "1" in the *SelectedQmCapabilities* input parameter, then the *QoS Manager* MUST provide the following information in the *ExtendedTrafficQosInfo* output argument to the CP:

1. UPnP-QoS traffic descriptor for all Blocking Streams along with their layer-2 stream information, if available, (QosSegmentId, Layer2StreamId);

2. Layer-2 stream information (QosSegmentId, Layer2StreamId) for all non-UPnP Blocking Streams, if available;

3. *UserImportanceNumber* for the TSPEC identified by *ActiveTspecIndex* of all the Blocking Streams and for the InitialTrafficDescriptor by invoking *GetListOfTrafficPolicies()* on the *QosPolicyHolder* Service identified in section 2.4.1.4.3 only if its version is 3.0.

If the *ExtendedTrafficQosInfo* output parameter is populated by the *QosManager*, then the admission is failed.

2.4.6.4.3.1.3.Requirements for Preemption:

If a Control Point sets the Preemption field to a value of "1" in the *SelectedQmCapabilities* input parameter and the version of the *QosPolicyHolder* Service identified in section 2.4.1.4.3 is 3.0 or higher, then the *QoS Manager* MUST perform the following steps :

1. If a Control Point sets the ReportBlockingStream field to a value of "0" in the *SelectedQmCapabilities* input parameter, and if all the Blocking Traffic Streams have higher or equal UINs for their TSPEC identified by ActiveTspecIndex, than the stream to be admitted or updated, then the *QoS Manager* MUST return Error Code 766 to the Control Point to indicate that preemption is not possible.

2. If a Control Point sets the ReportBlockingStream field to a value of "1" in the *SelectedQmCapabilities* input parameter, and if all the Blocking Traffic Streams have higher or equal UINs for their TSPEC identified by ActiveTspecIndex than the stream to be admitted or updated, then the *QosManager* Service MUST follow all the requirements identified in Section 2.4.6.4.3.1.2.

3. If some of the Blocking Traffic Streams have lower UINs for their TSPEC identified by ActiveTspecIndex than the stream to be admitted or updated, the *QoS Manager* takes into consideration characteristics of those Blocking Traffic Streams with lower UINs (such as resources utilization) to

determine the list of candidate traffic streams that need to be preempted. The actual method used by the *QoS Manager* to determine such a list of candidate traffic streams to be preempted is vendor-specific. Appendix B outlines some example methods. However, the *QoS Manager* MUST NOT preempt any Blocking Streams that have higher or equal UIN than the stream to be admitted or updated.

4. If the *QoS Manager* identifies that preemption of candidate traffic streams will not result in availability of sufficient resources for the stream to be admitted or updated on a QoS Segment, then the *QoS Manager* SHOULD NOT preempt any candidate traffic streams, and

    a. If the Control Point set the ReportBlockingStream field to a value of "0" in the *SelectedQmCapabilities* input parameter, the *QosManager* MUST return error to the Control Point with Error Code 766.

    b. If the Control Point set the ReportBlockingStream field to a value of "1" in the *SelectedQmCapabilities* input parameter, then the *QoS Manager* MUST follow all the requirements identified in section 2.4.6.4.3.1.2.

5. If the *QoS Manager* identifies that preemption of candidate traffic streams may result in the availability of sufficient resources for the stream to be admitted or updated, the *QoS Manager* MUST preempt each identified stream on all the *QosDevice* Services on the path of each stream by invoking *QD:ReleaseAdmittedQos()* (to revoke the resources allocated to that stream). The *QoS Manager* MUST provide the *PreemptingTrafficInfo* parameter as input argument of *QD:ReleaseAdmittedQos().*

6. After the *QoS Manager* completes preemption, it attempts to admit the new stream by invoking *QD:AdmitTrafficQos()* action on all the *QosDevice* Services in each of the Blocked QoS Segments.

7. If *QD:AdmitTrafficQos()* action is successful on all *QosDevice* Services in each of the Blocked Segments, the *QosManager* Service MUST return the *UpdatedTrafficDescriptor* to the Control Point with various traffic descriptor parameters set appropriately as per the requirements identified in Section 2.4.1.4.5. In this case, the QosManager Service MUST NOT populate the *ExtendedTrafficQosInfo* output parameter.

8. If *QD:AdmitTrafficQos()* fails on any one of the *QosDevice* Services on the path, the *QoS Manager* may retry preemption to admit the traffic stream. If retries are performed, the *QoS Manager* MUST use random backoff. If the *QoS Manager* determines that the traffic stream cannot be admitted after attempting preemption, then:

    a. If the Control Point set the ReportBlockingStream field to a value of "0" in the *SelectedQmCapabilities* input parameter, the *QosManager* MUST return error to the Control Point with Error Code 766.

    b. If the Control Point set the ReportBlockingStream field to a value of "1" in the *SelectedQmCapabilities* input parameter, then the *QoS Manager* MUST follow all the requirements identified in section 2.4.6.4.3.1.2.

While performing these additional capabilities of Preemption or ReporBlockingStream, in case of failure, the *QoS Manager* MUST release any resources that it might have reserved on any other QoS Segments for the traffic stream.

2.4.6.5. Dependency on State (if any)

Since UPnP *QosManager* is defined as a stateless entity, there is no effect of this action on state.

2.4.6.6. Effect on State (if any)

Since UPnP *QosManager* is defined as a stateless entity, there is no effect of this action on state.

2.4.6.7. Errors

Table 2-22: Error Codes for RequestExtendedTrafficQos

errorCode	errorDescription	Description
701	TrafficHandle must not be specified by the Control Point	TrafficHandle must not be specified by the Control Point
710	Incomplete TrafficId	At least one of the required parameters in the TrafficId is absent.
711	TrafficId could not be completed	The QosManager could not complete the TrafficId
713	Malformed Name-string	One or more of the String fields in the traffic descriptor do not conform to the specification.
715	Policy parameters must not be specified by the Control Point	PolicyLastModified, PolicyModifyingUserName, or PolicyHolderConfigUrl must not be specified by the Control Point
716	An input parameter (e.g. TrafficDescriptor) does not validate against the XML schema	One of the XML-based input arguments does not follow the schema
718	Invalid Requested QoS Type	Value supplied for RequestedQosType parameter is invalid
721	TrafficImportanceNumber must not be specified by the Control Point	TrafficImportanceNumber must not be specified by the Control Point
722	The ActiveTspecIndex must not be specified by the Control Point	The ActiveTspecIndex must not be specified by the Control Point
725	Duplicate TspecIndex	All Tspec indices must be unique
740	No QosDevice at Source IP address	Source QosDevice does not exist.
741	No QosDevice at Destination IP address	Destination QosDevice does not exist.
742	No QosDevice at QosBoundarySourceAddress	Source Boundary QosDevice does not exist.
743	No QosDevice at QosBoundaryDestinationAddress	Destination Boundary QosDevice does not exist.
744	No QosDevices available	No QosDevice instances found on the network.
746	QosPolicyHolder failure	QosManager gets incorrect information from the QosPolicyHolder (could be malformed XML)

errorCode	errorDescription	Description
752	QosBoundarySourceAddress or QosBoundaryDestinationAddress not on the same subnet	The source and/or destination is not on the subnet which is under the management of this QosManager
765	Admission Failure	A stream cannot be admitted due to lack of resources on the network.
766	Failure to admit even after preemption	Stream cannot be admitted even after preemption due to lack of resource
773	Cannot retrieve XML namespace	An XML namespace was included in one of the arguments, but the QosManager service was unable to retrieve the namespace from the location provided.
780	QosPolicyHolder not found	The QosPolicyHolder specified by the PolicyHolderId was not found.
781	Preferred QosPolicyHolder not found	The Preferred QosPolicyHolder not found on the network.
782	Parameterized QoS not possible	End to end parameterized QoS cannot be established for a stream due to presence of at least on prioritized QoS segment.
785	Different Preferred QosPolicyHolder ServiceId values	QosDevice Services report different ServiceId values for the Preferred QosPolicyHolder with the same highest preference count.
790	Capability not supported	The QosManager Service does not support the requested capability.
791	TrafficLeaseTime is missing	TrafficLeaseTime in the traffic descriptor is missing
792	Incorrect TSPEC	Input parameters supplied in the TSPEC are incorrect.

2.4.7. UpdateExtendedTrafficQos()

The Control Point invokes this optional action for updating QoS for a specified admitted traffic stream with optional additional capabilities, such as preemption and reporting of blocking streams that are not supported by *UpdateTrafficQos()*.

2.4.7.1. Arguments

Table 2-23: Arguments for UpdateExtendedTrafficQos

Argument	Direction	relatedStateVariable
TrafficHandle	In	A_ARG_TYPE_TrafficHandle
RequestedTrafficDescriptor	In	A_ARG_TYPE_TrafficDescriptor
SelectedQmCapabilities	In	A_ARG_TYPE_QmCapabilities
ImplementedTrafficDescriptor	Out	A_ARG_TYPE_TrafficDescriptor
ReturnTrafficQosInfo	Out	A_ARG_TYPE_ExtendedTrafficQosInfo

Argument	Direction	relatedStateVariable
ResultedQosType	Out	A_ARG_TYPE_ResultedQosType

2.4.7.2. Service Requirements

2.4.7.2.1. Base Requirements

Refer to Section 2.4.2.2.1 for base requirements when a Control Point invokes *UpdateExtendedTrafficQos()* action.

Note: The *QosManager* Service does not return NumPolicyHolders parameter as an output argument of *UpdateExtendedTrafficQos()* action.

2.4.7.2.2. RequestedQosType Specific Requirements

If a Control Point requests non-supported *QosManager* Service capability in the SelectedQmCapabilities input argument, the *QosManager* Service MUST return Error Code 790.

For *RequestedQosType* parameter with the value of "1" in the *RequestedTrafficDescriptor*, if the *QoS Manager* identifies that one of the QoS Segments on path is not capable of supporting Parameterized QoS, the *QosManager* Service MUST set *ResultedQosType* parameter to "1" which is returned as an output argument of *UpdateExtendedTrafficQos()* action. The *QoS Manager* determines whether a QoS Segment is capable of supporting parameterized QoS based on the *AdmitCntrlNet* parameter retrieved via the QD:*GetExtendedQosState()* action.

For *RequestedQosType* parameter with the value of "1" in the *RequestedTrafficDescriptor*, if the *QoS Manager* identifies that all of QoS Segments on path are capable of supporting Parameterized QoS, the *QosManager* Service MUST set *ResultedQosType* parameter to "2" which is returned as an output argument of *UpdateExtendedTrafficQos()* action.

For the *RequestedQosType* parameter with the value of "2" in the *RequestedTrafficDescriptor*, if the *QoS Manager* identifies that some of the QoS Segments on the path are not able to support Parameterized QoS, and no less preferred TSPEC is supplied, the *QosManager* Service MUST return error to the CP in response to *UpdateExtendedTrafficQos()* with Error Code of 782.

For the *RequestedQosType* parameter with the value of "1" or "2", if a Control Point does not supply a value for the *TrafficLeaseTime* field in the *RequestedTrafficDescriptor* input argument, the *QosManager* Service MUST return error with Error Code 791 to the Control Point.

2.4.7.3. Control Point requirements when calling the action

2.4.7.3.1. Base Requirements

Refer to Section 2.4.2.3.1 above when invoking *UpdatedExtendedTrafficQos()* action.

2.4.7.3.2. RequestedQosType Specific Requirements

A Control Point MUST provide *RequestedQosType* parameter in the *RequestedTrafficDescriptor* input argument.

A Control Point MUST NOT supply a value greater than "2" for the *RequestedQosType parameter.*

A Control Point MUST NOT request *QoS Manager* capabilities that are not supported by the *QosManager* Service.

For the *RequestedQosType* parameter with the value of "1" or "2", a Control Point MUST supply a value for the *TrafficLeaseTime* field in the *RequestedTrafficDescriptor* input argument.

2.4.7.4. QoS Manager Requirements

2.4.7.4.1. Base requirements

When a Control Point calls the *UpdatedExtendedTrafficQos()* action, the *QoS Manager* MUST follow all the requirements identified in Section 2.4.2.4.1, irrespective of the value of *RequestedQosType* parameter in the *InitialTrafficDescriptor* input parameter.

2.4.7.4.2. Prioritized QoS Update Requirements with Extended Functionality

Upon receiving *UpdateExtendedTrafficQos()* action with the *RequestedQosType* parameter in the *RequestedTrafficDescriptor* of "0", the *QoS Manager* MUST follow all the requirements identified in Section 2.4.2.4.2.

2.4.7.4.3. Parameterized and Hybrid QoS Update Requirements with Extended Functionality

Upon receiving *UpdateExtendedTrafficQos()* action with the *RequestedQosType* parameter in the *RequestedTrafficDescriptor* of either "1" or "2", the *QoS Manager* MUST follow all the requirements identified in Section 2.4.2.4.3 to update the stream identified by the *RequestedTrafficDescriptor*.

However, if the *QoS Manager* identifies that the stream cannot be updated even with the least preferred TSPEC supplied in the *RequestedTrafficDescriptor,* and if the Control Point requested extended *QoS Manager* capabilities (i.e. either Preemption = "1" or ReportBlockingStreams = "1" in the SelectedQmCapabilities input argument), the *QoS Manager* MUST perform the extended functions, as outlined in section 2.4.6.4.3.1 above, to update the stream with the least preferred TSPEC. If the CP did not request extended *QoS Manager* capabilities (i.e. CP set Preemption = "0" and ReportBlockingStreams = "0"), the *QoS Manager* MUST NOT perform the extended functions.

2.4.7.5. Dependency on State (if any)

Since *QosManager* is defined as a stateless entity, there is no dependency for this action on the state.

2.4.7.6. Effect on State (if any)

Since *QosManager* is defined as a stateless entity, there is no effect of this action on state.

2.4.7.7. Errors

Table 2-24: Error Codes for UpdateExtendedTrafficQos

errorCode	errorDescription	Description
703	Traffic Handle unknown to this device	Traffic Handle unknown to this device
713	Malformed Name-string	One or more of the String fields in the traffic descriptor do not conform to the specification.
714	Tried to update unmodifiable Traffic Descriptor elements	QosManager cannot be requested to update some of the TrafficDescriptor parameters e.g. TrafficId, TrafficHandle

errorCode	errorDescription Description	
715	Policy parameters must not be specified by the Control Point	PolicyLastModified, PolicyModifyingUserName, or PolicyHolderConfigUrl must not be specified by the Control Point
716	An input parameter (e.g. TrafficDescriptor) does not validate against the XML schema	One of the XML-based input arguments does not follow the schema
718	Invalid Requested QoS Type	Value supplied for RequestedQosType parameter is invalid
721	TrafficImportanceNumber must not be specified by the Control Point	TrafficImportanceNumber must not be specified by the Control Point
722	The ActiveTspecIndex must not be specified by the Control Point	The ActiveTspecIndex must not be specified by the Control Point
725	Duplicate TspecIndex	All Tspec indices must be unique
740	No QosDevice at Source IP address	Source QosDevice does not exist.
741	No QosDevice at Destination IP address	Destination QosDevice does not exist.
742	No QosDevice at QosBoundarySourceAddress	Source Boundary QosDevice does not exist.
743	No QosDevice at QosBoundaryDestinationAddress	Destination Boundary QosDevice does not exist.
744	No QosDevices available	No QosDevice instances found on the network.
746	QosPolicyHolder failure	QosManager gets incorrect information from the QosPolicyHolder (could be malformed XML)
752	QosBoundarySourceAddress or QosBoundaryDestinationAddress not on the same subnet	The source and/or destination is not on the subnet which is under the management of this QosManager
763	Update Failed	Failure in updating the traffic stream on one or more QosDevice Service(s) on the path.
766	Failure to update even after preemption	Stream cannot be updated even after preemption due to lack of resource
773	Cannot retrieve XML namespace	An XML namespace was included in one of the arguments, but the service was unable to retrieve the namespace from the location provided.
780	QosPolicyHolder not found	The QosPolicyHolder specified by the PolicyHolderId was not found.
781	Preferred QosPolicyHolder not found	The Preferred QosPolicyHolder was not found on the network.
782	Parameterized QoS not possible	End to end parameterized QoS cannot be established for a stream due to presence of at least on prioritized QoS segment.
785	Different Preferred QosPolicyHolder ServiceId values	QosDevice Services report different ServiceId values for the Preferred QosPolicyHolder with the same highest preference count.

errorCode	errorDescription	Description
790	Capability not supported	The QosManager Service does not support the requested capability.
791	TrafficLeaseTime is missing	TrafficLeaseTime in the traffic descriptor is missing
792	Incorrect TSPEC	Input parameters supplied in the TSPEC are incorrect.

2.4.8. Common Error Codes

The following table lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error MUST be returned. These common error codes have already been defined in UPnP Device Architecture and other Technical Committee documents.

Table 2-25: Common Error Codes

errorCode	errorDescription	Description
400-499		See UPnP Device Architecture section on Control.
500-599		See UPnP Device Architecture section on Control.
600-699		See UPnP Device Architecture section on Control.
701	TrafficHandle MUST NOT be specified by the Control Point	
703	Traffic Handle unknown to this device	
704	Network contains Different TrafficDescriptors for same TrafficHandle	
710	Incomplete TrafficId	At least one of the required parameters in the TrafficId is absent.
711	TrafficId could not be completed	The QosManager could not complete the TrafficId
713	Malformed Name-string	One or more of the String fields in the traffic descriptor do not conform to the specification.
714	Tried to update unmodifiable Traffic Descriptor elements	QosManager cannot be requested to update some of the TD parameters e.g. TrafficId, TrafficHandle
715	Policy parameters MUST NOT be specified by the Control Point	PolicyLastModified, PolicyModifyingUserName, or PolicyHolderConfigURL MUST NOT be specified by the Control Point

errorCode	errorDescription	Description
716	An input parameter (e.g. TrafficDescriptor) does not validate against the XML schema	One of the XML-based input arguments does not follow the schema
718	Invalid Requested QoS Type	Value supplied for RequestedQosType parameter is invalid
721	TrafficImportanceNumber MUST not be specified by the Control Point	
722	The ActiveTspecIndex MUST NOT be specified by the Control Point	
725	Duplicate TspecIndex	All Tspec indices MUST be unique
740	No QosDevice at Source IP address	Source QosDevice does not exist.
741	No QosDevice at Destination IP address	Destination QosDevice does not exist.
742	No QosDevice at QosBoundarySourceAddress	Source Boundary QosDevice does not exist.
743	No QosDevice at QosBoundaryDestinationAddress	Destination Boundary QosDevice does not exist.
744	No QosDevices available	No QosDevice instances found on the network.
746	QosPolicyHolder failure	QosManager gets incorrect information from the QosPolicyHolder (could be malformed XML)
752	QosBoundarySourceAddress or QosBoundaryDestinationAddress not on the QosManaged subnet	The source and/or destination is not on the subnet which is under the management of this QosManager
763	Update Failed	Failure in updating the traffic stream on one or more QosDevice Service(s) on the path.
765	Admission Failure	A stream cannot be admitted due to lack of resources on the network.
766	Failure to admit or update even after preemption	Stream cannot be updated even after preemption due to lack of resource
773	Cannot retrieve XML namespace	An XML namespace was included in one of the arguments, but the service was unable to retrieve the namespace from the location provided.

errorCode	errorDescription	Description
780	QosPolicyHolder not found	The QosPolicyHolder specified by the PolicyHolderId was not found.
781	Preferred QosPolicyHolder not found	The Preferred QosPolicyHolder was not found on the network.
782	Parameterized QoS not possible	End to end parameterized QoS cannot be established for a stream due to presence of at least on prioritized QoS segment.
785	Different Preferred QosPolicyHolder ServiceId values	QosDevice Services report different ServiceId values for the Preferred QosPolicyHolder with the same highest preference count.
790	Capability not supported	The QosManager Service does not support the requested capability.
791	TrafficLeaseTime is missing	TrafficLeaseTime in the traffic descriptor is missing
792	Incorrect TSPEC	Input parameters supplied in the TSPEC are incorrect.

2.5. Theory of Operation (Informative)

2.5.1. Overview

The *QosManager* is responsible for setting up, updating, revoking and controlling the QoS assigned by networking devices to various traffic streams. In the context of UPnP AV, the AV Control Point invokes the *QosManager* to perform the functions related to setting up QoS for a particular traffic. In case of non- UPnP-AV scenario, any application (acting as a Control Point) can invoke the *QosManager* service for setting up the QoS for a particular traffic.

There are three types of QoS requests handled by the *QosManager*. A Control Point may request Prioritized QoS for streams. Streams with no explicit priority default to the best-effort priority. A Control Point may also request Parameterized QoS to actually reserve some resources on the network for the exclusive use of a stream. Note: Networks may recover unused reserved resources to give to other network traffic. In some cases a Control Point may request Parameterized QoS but it may not be available on all segments of the network. If the Control Point indicates that a hybrid path containing both Parameterized and Prioritized segments is acceptable, the *QosManager* sets up the stream with Parameterized QoS on some v3 *QosDevice* services and Prioritized QoS on v2 or older *QosDevice* services.

The *QosManager* requests allocation of resources from all *QosDevice* Services along the path of the stream. Each *QosDevice* is responsible for the allocation of resources on the segments that it is connected to. The actual responsibility for allocation will vary depending on the technology of the lower layer network attached to each *QosDevice*. Some physical layers initiate the request for resource allocattion from the source end of the stream, some from the sink end, and some need both ends to be informed of the need for QoS. The *QosManager* is not aware of the needs of each physical layer. By calling the various actions on all QDs in the path it ensures that all three of the above scenarios are covered.

A *QosManager* acts as a control point to UPnP *QosDevice* services. A *QosManager* discovers the *QosDevice* services on the local area network, and may use *QD:GetExtendedQosState()* actions exposed by the *QosDevice* service to get information about the device. *QD:GetExtendedQosState()* action returns the current state of the UPnP-QoS enabled device with information such as the TrafficDescriptors that are currently active.

A Control Point may discover information on all registered streams on the network by calling the *QM:BrowseAllTrafficDescriptors()* action. The *QosManager* responds with a list of Traffic Descriptors for all of the streams on the network.

It should be noted that UPnP-QoS defines services (*QosManager*, *QosDevice*, and *QosPolicyHolder*) but it does not define a new device type. Since the QoS problem needs to be solved across the board for multiple usage scenarios, it is expected that vendors may use any UPnP device as a container for the services defined in UPnP-QoS. The control points and *QoS Managers* should search for UPnP-QoS services embedded in all UPnP device types.

2.5.1.1. Preemption of Existing Streams

There are many application scenarios where it is possible and even desirable to provide a way to admit a stream when there are not enough resources on all or part of the stream's path through the network. Examples are an incoming phone call, or a scheduled DVR recording. UPnP-QoS v3 provides a method to preempt existing reservations and allow for the admission of these streams following the policies provided by the preferred *QosPolicyHolder* Service.

When an admission request fails due to the lack of bandwidth or other resources, the *QosManager* may attempt to free resources necessary to allow the new admission. All existing streams on QoS Segments that are using

resources needed to admit a new stream may be candidates for preemption (Blocking Streams[6]). The *QosManager* calls the *QosPolicyHolder* with the *QPH:GetListOfTrafficPolicies()* action to rank importance of the streams on a segment. The *QosPolicyHolder* returns a list of UserImportanceNumbers (UINs) to provide a relative rating of the streams. The *QosManager* may preempt (remove or downgrade) the QoS for the blocking streams with lower UIN than the new stream at all QosSegments on the path of those streams. After the preemption, *QosManager* attempts admission of the stream again.

The *QosManager* also provides information about Blocking Streams along with their UserImportanceNumbers (UINs) to a Control Point to allow it to manipulate the streams directly (e.g. Preemption).

2.5.2. Recovery from Failures

Reservations in UPnP-QoS v3 result in the allocation of finite network resources. When a stream or an application no longer needs the resources they must be freed so that other streams or applications may use them. Failure of a Control Point, stream, application, or network part, may result in the failure to free some or all of the resources allocated and reserved for that stream. To prevent the orphaning of resources and the eventual degradation of network performance and capability it is necessary for mechanisms to be in place that will detect the absence of the normal release process and free the resources.

A lease time parameter is provided to indicate to the *QosDevice* the time period for which the reservation is to be active. If a timer set to the indicated lease time expires with no explicit release of the reservation, the *QosDevice* may then assume that the reservation is no longer needed, release the reservation and free the resources.

2.5.2.1. Example 1

The suggested method for using the timer is that a Control Point will reserve bandwidth for a short period of time (seconds or minutes depending on the needs of the application), initiate the stream, and then refresh the reservation before the timer expires to maintain the resource allocation. In this way any failure will result in the lease timers expiring at the least time interval and the allocated resources will be recovered. This is the expected behavior for a Control Point that resides with either the source or sink of a stream.

2.5.2.2. Example 2

In some cases the initiating Control Point may be in a third party device, such as a remote control, and may not continue to be active after the reservation and stream have been set up. In this case the Control Point may set a long lease time. This lease time would be enough for the entire event. For example, a Control Point may initiate the turning on of a TV from a remote. The expected behavior of a TV is that it stays on until someone turns it off. In this case the remote may set a very long lease time for a particular event. The disadvantage of this method comes if there is some failure in either the network or in the Source or Sink device. In this situation resources remain reserved on the network even though they are not being used and they will not be freed until the, potentially lengthy, lease timer expires.

As an optimization to prevent a long lease time from leaving the resources allocated in a failure, the *QosDevice* on the TV could detect a long lease time, and after the stream is started do a refresh with a short lease time. This then meets the criteria of having a short recovery time and maintains that lease until someone actually turns off the TV.

For failure recovery to work efficiently lease times should be relatively short. Specific lease times depend on the needs of the applications. Refresh times should be ½ to 1/3 the lease time to allow for dropped refresh messages.

Request, Update and Release actions are sent to all *QosDevice*ss in the path by the *QosManager*. This ensures that all devices have the same lease time for a particular stream.

---

[6] See the 'Terminology' section.

3. XML Service Description

```
<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <actionList>
      <action>
         <name>RequestTrafficQos()</name>
         <argumentList>
            <argument>
               <name>InitialTrafficDescriptor</name>
               <direction>in</direction>
               <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor
               </relatedStateVariable>
            </argument>
            <argument>
               <name>TrafficHandle</name>
               <direction>out</direction>
               <relatedStateVariable>A_ARG_TYPE_TrafficHandle
               </relatedStateVariable>
            </argument>
            <argument>
               <name>NumPolicyHolders</name>
               <direction>out</direction>
               <relatedStateVariable>A_ARG_TYPE_NumPolicyHolders
               </relatedStateVariable>
            </argument>
            <argument>
               <name>UpdatedTrafficDescriptor</name>
               <direction>out</direction>
               <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>BrowseAllTrafficDescriptors</name>
         <argumentList>
            <argument>
               <name>NumberOfTrafficDescriptors</name>
               <direction>out</direction>
               <relatedStateVariable>A_ARG_TYPE_NumTrafficDescriptors
               </relatedStateVariable>
            </argument>
            <argument>
               <name>TrafficDescriptorList</name>
               <direction>out</direction>
               <relatedStateVariable>A_ARG_TYPE_ListOfTrafficDescriptors
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>ReleaseTrafficQos</name>
         <argumentList>
            <argument>
               <name>RevokeTrafficHandle</name>
               <direction>in</direction>
               <relatedStateVariable>A_ARG_TYPE_TrafficHandle
               </relatedStateVariable>
            </argument>
```

```
      </argumentList>
   </action>
   <action>
      <name>UpdateTrafficQos</name>
      <argumentList>
         <argument>
            <name>TrafficHandle</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficHandle
            </relatedStateVariable>
         </argument>
         <argument>
            <name>RequestedTrafficDescriptor</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor
            </relatedStateVariable>
         </argument>
         <argument>
            <name>ImplementedTrafficDescriptor</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor
            </relatedStateVariable>
         </argument>
         <argument>
            <name>NumPolicyHolders</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_NumPolicyHolders
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>
   <action>
      <name>RequestExtendedTrafficQos</name>
      <argumentList>
         <argument>
            <name>InitialTrafficDescriptor</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor
            </relatedStateVariable>
         </argument>
         <argument>
            <name>SelectedQmCapabilities</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_QmCapabilities
            </relatedStateVariable>
         </argument>
         <argument>
            <name>TrafficHandle</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficHandle
            </relatedStateVariable>
         </argument>
         <argument>
            <name>UpdatedTrafficDescriptor</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor
            </relatedStateVariable>
         </argument>
         <argument>
            <name>ExtendedTrafficQosInfo</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_ExtendedTrafficQosInfo
            </relatedStateVariable>
         </argument>
         <argument>
            <name>ResultedQosType </name>
```

```
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_ResultedQosType
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>
   <action>
      <name>UpdateExtendedTrafficQos</name>
      <argumentList>
         <argument>
            <name>TrafficHandle</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficHandle
            </relatedStateVariable>
         </argument>
         <argument>
            <name>RequestedTrafficDescriptor</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor
            </relatedStateVariable>
         </argument>
         <argument>
            <name>SelectedQmCapabilities</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_QmCapabilities
            </relatedStateVariable>
         </argument>
         <argument>
            <name>ImplementedTrafficDescriptor</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor
            </relatedStateVariable>
         </argument>
         <argument>
            <name>ExtendedTrafficQosInfo</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_ExtendedTrafficQosInfo
            </relatedStateVariable>
         </argument>
         <argument>
            <name>ResultedQosType</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_ResultedQosType
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>
   <action>
      <name>GetQmCapabilities</name>
      <argumentList>
         <argument>
            <name>QmCapabilities</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_QmCapabilities
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>
   Declarations for other actions added by UPnP vendor (if any) go here
</actionList>
<serviceStateTable>
   <stateVariable sendEvents="No">
      <name>A_ARG_TYPE_NumPolicyHolders</name>
      <dataType>ui4</dataType>
   </stateVariable>
   <stateVariable sendEvents="No">
```

```
         <name>A_ARG_TYPE_NumTrafficDescriptors</name>
         <dataType>ui4</dataType>
      </stateVariable>
      <stateVariable sendEvents="No">
         <name>A_ARG_TYPE_TrafficHandle</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="No">
         <name>A_ARG_TYPE_TrafficDescriptor</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="No">
         <name>A_ARG_TYPE_ListOfTrafficDescriptors</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="No">
         <name>A_ARG_TYPE_QmCapabilities</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="No">
         <name>A_ARG_TYPE_ExtendedTrafficQosInfo</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="No">
         <name>A_ARG_TYPE_ResultedQosType</name>
         <dataType>ui4</dataType>
      </stateVariable>
      Declarations for other state variables added by UPnP vendor (if any) go here
   </serviceStateTable>
</scpd>
```

4. Test

No semantic tests have been specified for this service.

Appendix A. Traffic Descriptor Matrix

Parameter Definition		Populated By			Used By[7]			Comments
Parameter Name	**R/O**	**CP**	**QM**	**QD**	**CP**	**QM**	**QD**	
TrafficHandle	R		*		*	*	*	
TrafficId	R							
. SourceAddress	O	*	*			*	*	A value of 0.0.0.0 represents 'Don't Care' i.e. condition where Control Point wants to set up QoS for all traffic streams irrespective of the SourceAddress.
. SourcePort	O	*	*			*	*	If SourcePort tag is missing, it represents value not known. This may happen in a '3 box model'.[8] A value of 0 represents 'Don't Care' i.e. condition where Control Point wants to set up QoS for all traffic streams irrespective of the SourcePort.
. DestinationAddress	O	*	*			*	*	A value of 0.0.0.0 represents 'Don't Care' i.e. condition where Control Point wants to set up QoS for all traffic streams irrespective of the DestinationAddress.
. DestinationPort	O	*	*			*	*	If DestinationPort tag is missing, it represents value not known. This may happen in a '3 box model'.[9] A value of 0 represents 'Don't Care' i.e. condition where Control Point wants to set up QoS for all traffic streams irrespective of the DestinationPort.
. IpProtocol	O	*	*[10]			*	*	If Protocol tag is missing, it represents value not known. A value of 0 represents 'Don't Care' i.e. condition where Control Point wants to set up QoS for all traffic streams irrespective of the Protocol.
. v2TrafficId	O							
. . v3TrafficId	O							

[7] Policy Holder may use any or all parts of the TD to make policy decisions. SincePolicy Holder is proprietary and policy definition is out of band for UPnP-QoS; this table does not define what TD parameters are 'used by'Policy Holder.

[8] Please refer to the UPnP QoS Architecture document for more information around '3 box model'.

[9] Please refer to the UPnP QoS Architecture document for more information around '3 box model'.

[10] *QosManager* may query *QD:GetQosDeviceInfo()* action call to obtain the IP Protocol.

Parameter Definition		Populated By			Used By[7]			Comments
Parameter Name	**R/O**	**CP**	**QM**	**QD**	**CP**	**QM**	**QD**	
. . . SourceUuid	O	*				*		
. . . DestinationUuid	O	*				*		
AvailableOrderedTspecList	R							
. Tspec	O							
. . TspecIndex	R	*				*	*	
. . AvTransportUri	O	*				*	*	
. . AvTransportInstanceId	O	*				*	*	
. . TrafficClass	R	*				*	*	
. . v2TrafficSpecification	O							
. . . v3TrafficSpecification	O							
. . . . RequestedQosType	O	*				*	*	
. . . . DataRate	O	*				*	*	
. . . . MaxBurstSize	O	*				*	*	
. . . . MinServiceRate	O	*				*	*	
. . . . ReservedServiceRate	O	*				*	*	
. . . . TimeUnit	O	*				*	*	
. . . . MaxPacketSize	O	*				*	*	
. . . . E2EMaxDelayHigh	O	*				*	*	
. . . . E2EMaxDelayLow	O	*				*	*	
. . . . E2EMaxJitter	O	*				*	*	
. . . . QosSegmentSpecificParameters	O							
. . . . . InterfaceId	O		*				*	
. . . . . QosSegmentId	O		*				*	
. . . . . QosSegmenMaxDelayHigh	O		*				*	
. . . . . QosSegmenMaxDelayLow	O		*				*	
. . . . . QosSegmenMaxDelayJitter	O		*				*	
. . . . MaxServiceInterval	O	*				*	*	
. . . . MinServiceInterval	O	*				*	*	
. . . . LossSensitivity	O	*				*	*	
. . . . LossInterval	O	*				*	*	
. . . . LossSensitivity	O	*				*	*	
. . . . ServiceType	O	*				*	*	
ActiveTspecIndex	R		*		*	*	*	
TrafficImportanceNumber	R		*				*	This information may come from QosPolicyHolder Service, but is populated in TD by QosManager.

Parameter Definition		Populated By			Used By[7]			Comments
Parameter Name	**R/O**	**CP**	**QM**	**QD**	**CP**	**QM**	**QD**	
QosBoundarySourceAddress	O	*				*	*	
QosBoundaryDestinationAddress	O	*				*	*	
MediaServerConnectionId	O	*				*	*	
MediaRendererConnectionId	O	*				*	*	
TrafficLeaseTime	O	*				*	*	
PolicyHolderId	O	*				*		If the Control Point wants to specify a specific QosPolicyHolder to be used
PolicyLastModified	O		*		*			Obtained from QosPolicyHolder Service.
PolicyModifyingUserName	O		*		*			Obtained from QosPolicyHolder Service.
PolicyHolderConfigUrl	O		*		*			Obtained from QosPolicyHolder Service.
QosBoundarySourceUuid	O	*				*	*	
QosBoundaryDestinationUuid	O	*				*	*	
Critical	O	*				*		
PolicyHolderConsultedId	O		*		*			
PolicyHolderConsultedType	O		*		*			
OptionalPolicyParams	O							
UserName	O	*			*			
VendorApplicationName	O	*			*			
PortName	O	*			*			
ServiceProviderServiceName	O	*			*			
CpName	O	*			*			

Appendix B. (Informative) Methods for determining candidate streams for preemption based on UINs

The following are a few example methods for determing candidate streams for preemption based on UINs. Method # c is recommended:

a. Delete/downgrade blocking streams from the lowest UIN up to the UIN of the preempting stream until sufficient resources are available.

b. Delete/downgrade the minimum number of streams with UINs lower than the preempting stream so that sufficient resources are available for the preempting stream.

c. Find the smallest UIN x (at most the UIN of a preempting stream) for which deletion/downgrade of all streams with UIN at most x provide sufficient resources. Then delete/downgrade the minimum number of streams with UIN less than x that provide sufficient resources for the preempting stream.

Example: The necessary bandwidth for the new stream is 10Mbps with UIN 50, and there are blocking streams as follows.

5Mbps (UIN=10)

3Mbps (UIN=20)

5Mbps (UIN=30)

10Mbps (UIN=40)

Results using Method #a: Pre-empted streams: 5Mbps (UIN=10) + 3Mbps (UIN=20) + 5Mbps (UIN=30)-> 13Mbps

Results using Method #b: Pre-empted streams: 10Mbps (UIN=40) -> 10Mbps

Results using Method #c: UIN x = 30, Pre-empted streams: 5Mbps (UIN=10) + 5Mbps (UIN=30) -> 10Mbps

Appendix C. (Normative) The *res@tspec* property in the *ContentDirectory* Service

The UPnP AV *ContentDirectory:2* Service introduces the *res@tspec* property as follows and as interpreted. "The *res@tspec* property identifies the content's QoS characteristics. The *res@tspec* property has a maximum length of 256 characters. The details about this property, including its components and formatting constraints, are defined in the *QosManager* Service definition document". This Appendix contains those details.

The *res@tspec* property is a string of which the first four characters are "<" followed by an escaped XML fragment. Otherwise it has to be treated as a URI as defined in [RFC 3986]. The indirection through a URI is provided to bound the length of the string without restricting the number of TSPEC elements that can be provided. If the length of the string with the value for res@tspec property would otherwise exceed 256 characters the URI format MUST be used. If the length of the string remains smaller than 256 characters, it is recommended to directly include the escaped XML fragment.

If the *res@tspec* property contains an escaped XML fragment, then the unescaped XML fragment MUST be as specified in Section C.1.

If the *res@tspec* property contains a URI then HTTP-GET MUST be used to retrieve the XML fragment as specified in Section C.1.

C.1 Definition of the XML Fragment

The XML fragment in this argument MUST start with a root tag which is of type "`TrafficSpecificationType`". This type is defined in the XML schema for the namespace "`http://www.upnp.org/schemas/TrafficDescriptorv1.xsd`". The XML schema is located at "http://www.upnp.org/schemas/qos/TrafficDescriptor-v3.xsd".

Example:

```
<tspec>
      <DataRate>10000000</DataRate>
      <PeakDataRate>10000000</PeakDataRate>
</tspec>
```