ISO/IEC 29341-17-10

Edition 1.0 2011-09

INTERNATIONAL STANDARD

colour inside

Information technology – UPnP device architecture – Part 17-10: Quality of Service Device Control Protocol – Level 3 – Quality of Service Device Service

ISO/IEC 29341-17-10:2011(E)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

ISO/IEC 29341-17-10

Edition 1.0 2011-09

INTERNATIONAL STANDARD

colour inside

Information technology – UPnP device architecture – Part 17-10: Quality of Service Device Control Protocol – Level 3 – Quality of Service Device Service

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE X

ICS 35.200

ISBN 978-2-88912-657-6

CONTENTS

INFORMATION TECHNOLOGY – UPNP DEVICE ARCHITECTURE –

Part 17-10: Quality of Service Device Control Protocol – Level 3 – Quality of Service Device Service

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.

4) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.

6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC publication.

7) All users should ensure that they have the latest edition of this publication.

8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.

9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

10) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 29341-17-10 was prepared by UPnP Forum Steering committee[1], was adopted, under the fast track procedure, by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

The list of all currently available parts of the ISO/IEC 29341 series, under the general title *Information technology – UPnP device architecture*, can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies, and the voting results may be obtained from the address given on the second title page.

---

1 UPnP Forum Steering committee, UPnP Forum, 3855 SW 153rd Drive, Beaverton, Oregon 97006 USA. See also “Introduction”.

IMPORTANT – The "colour inside" logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this publication using a colour printer.

1 Overview and Scope

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

This service-type enables modeling of the 'QosDevice' function capabilities. The QosDevice:3 Service is a function typically implemented in source, sink and intermediate network. The *QosDevice* Service is responsible for providing the appropriate network resources to traffic streams and information about the state of the device as requested by the QosManager as defined in the QosManager:3 Service. [QM]

Several L2 Technologies were considered during the design of UPnP-QoS v3. These technologies are described in UPnP QosDevice:3 Underlying Technology Interface Addendum [QD_Add] . Every attempt was made to ensure that the design of version 3 would accommodate other L2 Technologies as well. Each L2 Technology on which UPnP-QoS version 3 is implemented is recommended to have a document that is compliant to the template in Annex B which specifies how the L2 Technology defines certain state variables, maps parameters, etc.

This document does not address the procedures for end-to-end set up of a new traffic stream or end-to-end revocation of an existing traffic stream. This procedure is defined in the UPnP QosManager:3 Service Document [QM] .

1.1 Referenced Specifications

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.

1.1.1 Normative References

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

[Annex_G] – IEEE 802.1D-2004, Annex G, IEEE Standard for Information technology - Telecommunications and information exchange between systems - IEEE standard for local and metropolitan area networks - Common specifications - Media access control (MAC) Bridges, 2004.

[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.

[QM] – UPnP QosManager:3 Service Document: This reference is informative except for the definitions of the following state variables, which are normative: *A_ARG_TYPE_TrafficDescriptor*, *A_ARG_TYPE_NumTrafficDescriptors* and *A_ARG_TYPE_TrafficHandle*.
Available at: http://www.upnp.org/specs/qos/UPnP-qos-QosManager-v3-Service-20081130.pdf
Latest version available at: http://www.upnp.org/specs/qos/UPnP-qos-QosManager-v3-Service.pdf

[QPH] - *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

[DEVICE] - *UPnP Device Architecture, version 1.0*.

[RFC3339] – Date and Time on the Internet: Timestamps, G. Klyne, July 2002. http://www.ietf.org/rfc/rfc3339.txt

[IANA] - IANA Interface Type (IANAifType)-MIB http://www.iana.org/assignments/ianaiftype-mib

[QD_Add] –*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

1.1.2 Informative References

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

[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

[HomePlug AV] – HomePlug AV Specification, version 1.1.00, Homeplug Powerline Alliance, www.HomePlug.org.

[MoCA1.0] MoCA MAC/PHY SPECIFICATION v1.0, 2006.

[MoCA1.1] MoCA MAC/PHY SPECIFICATION v1.1 EXTENSIONS. 2007.

[IEEE802.3] – IEEE Standard for Information technology— Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications IEEE Std 802.3™-2005. http://standards.ieee.org/getieee802/802.3.html

[IEEE11] - 802.11-2007 IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks— Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications http://shop.ieee.org/ieeestore/Product.aspx?product_no=SS95708

[DSCP] - IETF RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, K. Nichols et al., December 1998. Available at: http://www.ietf.org/rfc/rfc2474.txt

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:*QosDevice:3*

The term *QosDevice* Service is used herein to refer to this type of service.

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] Clause 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	R/O [a]	Data Type	Allowed Value [b]	Default Value [b]	Eng. Units
A_ARG_TYPE_TrafficDescriptor	R	String (XML fragment)	See 2.2.2	n/a	n/a
A_ARG_TYPE_TrafficDescriptorContainer	R	String (XML fragment)	See 2.2.31	n/a	n/a
A_ARG_TYPE_TrafficDescriptorsPerInterface	R	String (XML fragment)	See 2.2.3	n/a	n/a
A_ARG_TYPE_TrafficHandle	R	**string**	See 2.2.4	n/a	n/a
A_ARG_TYPE_NumTrafficDescriptors	R	ui4	See 2.2.5	n/a	n/a
A_ARG_TYPE_QosDeviceCapabilities	R	String (XML fragment)	See 2.2.6	n/a	n/a
A_ARG_TYPE_QosDeviceState	R	String (XML fragment)	See 2.2.7	n/a	n/a
PathInformation	R	String (XML fragment)	See 2.2.8	n/a	n/a
A_ARG_TYPE_QosDeviceInfo	R	String (XML fragment)	See 2.2.9	n/a	n/a
A_ARG_TYPE_QosDeviceInfoContainer	R	String (XML fragment)	See 2.2.33	n/a	n/a
A_ARG_TYPE_QosStateId	R	**string**	See 2.2.10	n/a	n/a
A_ARG_TYPE_NumRotameterObservations	O	ui4	See 2.2.11	1	n/a
A_ARG_TYPE_RotameterInformation	O	String (XML fragment)	See 2.2.12	n/a	n/a
A_ARG_TYPE_ConfRotameterObservations	O	String (XML fragment)	See 2.2.13	n/a	n/a

Variable Name	R/O [a]	Data Type	Allowed Value [b]	Default Value [b]	Eng. Units
MostRecentStreamAction	O	String (XML fragment)	See 2.2.14	n/a	n/a
A_ARG_TYPE_MaxPossibleRotameterObservations	O	ui4	See 2.2.15	1	n/a
A_ARG_TYPE_Resource	R	String (XML fragment)	See 2.2.16	n/a	n/a
A_ARG_TYPE_AdmitTrafficQosExtendedResult	R	String (XML fragment)	See 2.2.17	n/a	n/a
A_ARG_TYPE_ListOfAdmittedTraffic	R	String (XML fragment)	See 2.2.18	n/a	n/a
A_ARG_TYPE_PreferredQph	O	String (XML fragment)	See 2.2.19	n/a	n/a
UnexpectedStreamChange	R	ui4	See 2.2.20	n/a	n/a
A_ARG_TYPE_PreemptingTrafficInfo	O	String (XML fragment)	See 2.2.21	n/a	n/a
A_ARG_TYPE_ListOfMostRecentUnexpectedStreamChanges	O	String (XML fragment)	See 2.2.22	n/a	n/a
A_ARG_TYPE_QosDeviceExtendedState	R	String (XML fragment)	See 2.2.23	n/a	n/a
A_ARG_TYPE_Layer2Mapping	R	String (XML fragment)	See 2.2.24	n/a	n/a
A_ARG_TYPE_Layer2MappingContainer	R	String (XML fragment)	See 2.2.32	n/a	n/a
A_ARG_TYPE_AdmitTrafficQosSucceeded	R	**boolean**	See 2.2.25	n/a	n/a
A_ARG_TYPE_TrafficDescriptorsWanted	R	**boolean**	See 2.2.26	n/a	n/a
A_ARG_TYPE_SetPreferredQphResults	O	ui4	See 2.2.27	n/a	n/a
A_ARG_TYPE_NumberOfUnexpectedStreamChangesRequested	O	ui4	See 2.2.28	n/a	n/a
A_ARG_TYPE_NumberOfUnexpectedStreamChangesReported	O	ui4	See 2.2.29	n/a	n/a
A_ARG_TYPE_NewTrafficLeaseTime	R	ui4	See 2.2.30	n/a	n/a

a R = Required, O = Optional, X = Non-standard

b 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

This required state variable is defined in the *QosManager* Service specification; it contains QoS related information for a traffic stream. Refer to [QM] for details of this state variable.

2.2.2.1 XML Schema Definition

This is a **string** containing an XML fragment. It contains information describing the 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"`.

2.2.3 A_ARG_TYPE_TrafficDescriptorsPerInterface

This required state variable contains the list of traffic descriptors that are associated with a network interface on a given *QosDevice* Service.

2.2.3.1 XML Schema Definition

This is a **string** containing an XML fragment. The XML fragment in this argument MUST validate against the XML schema for `TrafficDescriptorsPerInterface` in the XML namespace `"http://www.upnp.org/schemas/TrafficDescriptorsPerInterface.xsd"` which is located at "`http://www.upnp.org/schemas/qos/TrafficDescriptorsPerInterface-v2.xsd`".

2.2.3.2 Description of fields in the TrafficDescriptorsPerInterface structure

The *TrafficDescriptorsPerInterface* is a structure that consists of one or more entries of *TdInterfacePair*. *TdInterfacePair* lists one *TrafficDescriptor*, followed by the *InterfaceId* of the associated interface. Here are the details about these two parameters:

TrafficDescriptor: This required field describes a Traffic Descriptor associated with an Interface. An Interface can have multiple associated Traffic Descriptor objects.

InterfaceId: This is a required field of type **string**; its format is defined in clause 2.2.6.2.

2.2.3.3 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<TrafficDescriptorsPerInterface
xmlns="http://www.upnp.org:qos:tdpi2/schemas/TrafficDescriptorsPerInterface.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/TrafficDescriptorsPerInterface.xsd
http://www.upnp.org/schemas/qos/TrafficDescriptorsPerInterface-v2.xsd">
   <TdInterfacePair>
      <TrafficDescriptor>
         <td:TrafficHandle>wxyz</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>
            <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>
         <td:ActiveTspecIndex>300</td:ActiveTspecIndex>
         <td:TrafficImportanceNumber>5</td:TrafficImportanceNumber>
         <td:OptionalPolicyParams>
            <td:CpName>Amy's CP</td:CpName>
         </td:OptionalPolicyParams>
      </TrafficDescriptor>
      <InterfaceId>eth0</InterfaceId>
   </TdInterfacePair>
</TrafficDescriptorsPerInterface>
```

2.2.4 A_ARG_TYPE_TrafficHandle

A_ARG_TYPE_TrafficHandle is a **string** to identify a traffic stream. Refer to the [QM] document for more details.

2.2.5 A_ARG_TYPE_NumTrafficDescriptors

This is an integer argument specifying the number of Traffic Descriptors contained in the accompanying *ListOfTrafficDescriptors*. Refer to the [QM] document for more details.

2.2.6 A_ARG_TYPE_QosDeviceCapabilities

This required structure contains information describing a device's QoS capabilities. Use of this state variable is discouraged for UPnP-QoS v3. For v3 *QosDevice* Services, the information contained in this state variable can also be found in the *A_ARG_TYPE_QosDeviceExtendedState* state variable.

2.2.6.1 XML Schema Definition

This is a **string** containing an XML fragment. It contains information describing the capabilities of the *QosDevice* Service. The XML fragment in this argument MUST validate against the XML schema for `QosDeviceCapabilities` in the XML namespace "`http://www.upnp.org/schemas/QosDeviceCapabilities.xsd`" which is located at "`http://www.upnp.org/schemas/qos/QosDeviceCapabilities-v2.xsd`".

2.2.6.2 Description of fields in the QosDeviceCapabilities structure

Interface: This is a required structure and defined as an XML element. This field describes a network interface on the *QosDevice* Service. An Interface definition is required for each interface supported by the device. This information is provided even if the physical interface is down at a given time.

MacAddress: This is an optional field. If a given interface has an associated MAC address, the QosDevice MUST provide this information here. It provides the MAC address of the Interface and is of type MacAddressType (defined in the schema).

InterfaceId: This is a required field. The value is of type **string** and MUST uniquely identify an interface within the *QosDevice* Service. Furthermore, the *InterfaceId* MUST remain the same for a given interface (L2 Technology) until the *QosDevice* Service reboots.

IanaTechnologyType: This is an optional integer field. The *IanaTechnologyType* (IANA uses the designation IANAifType) is an integer assigned by IANA for any media type, such as a value of 6 for 802.3 media type or a value of 71 for 802.11 media type The allowed integer values for this parameter are specified in [IANA].

AdmissionControlSupported: This is a required enumeration field. This field is maintained for backward compatibility. This field can report only one of two values "Yes" or "No".. QosManager:3 ignores the value of this field.

PacketTaggingSupported: This is a required enumeration field. *PacketTaggingSupported* field indicates whether the device is capable of tagging L2 priorities on the outgoing interface. This field can report only one of two values "Yes" or "No".

NativeQos: This is an optional enumeration field. To ensure backward compatibility, this field MUST contain one of the values (Prioritized, BestEffort).

MaxPhyRate: Indicates the maximum PHY rate of the interface and expressed as a value of type unsignedInt. This parameter is required and indicates (Units) phy rate measured in bits/sec.

ChannelInformation: This is an optional unsignedInt field . It indicates the channel number of the *IanaTechnologyType* (IANAifType), if the technology supports channels. For example, 802.11 (value=71) supports multiple channels. Expressed as a value of type UnsignedInt.

2.2.6.3 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<QosDeviceCapabilities
xmlns="http://www.upnp.org/schemas/QosDeviceCapabilities.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/QosDeviceCapabilities.xsd
http://www.upnp.org/schemas/qos/QosDeviceCapabilities-v2.xsd">
  <Interface>
    <InterfaceId>eth0</InterfaceId>
    <MacAddress>0212abcdef11</MacAddress>
    <IanaTechnologyType>6</IanaTechnologyType>
    <AdmissionControlSupported>No</AdmissionControlSupported>
    <PacketTaggingSupported>Yes</PacketTaggingSupported>
    <NativeQos>Prioritized</NativeQos>
    <MaxPhyRate>100000000</MaxPhyRate>
  </Interface>
  <Interface>
    <InterfaceId>eth1</InterfaceId>
    <MacAddress>0212abcdef12</MacAddress>
    <IanaTechnologyType>71</IanaTechnologyType>
    <AdmissionControlSupported>No</AdmissionControlSupported>
    <PacketTaggingSupported>Yes</PacketTaggingSupported>
    <NativeQos>Prioritized</NativeQos>
    <MaxPhyRate>3000000</MaxPhyRate>
    <v2>
      <ChannelInformation>6</ChannelInformation>
    </v2>
  </Interface>
  <Interface>
    <InterfaceId>eth2</InterfaceId>
    <MacAddress>0212abcdef13</MacAddress>
    <IanaTechnologyType>6</IanaTechnologyType>
    <AdmissionControlSupported>No</AdmissionControlSupported>
    <PacketTaggingSupported>Yes</PacketTaggingSupported>
    <NativeQos>BestEffort</NativeQos>
    <MaxPhyRate>5000000</MaxPhyRate>
  </Interface>
  <Interface>
    <InterfaceId>example1</InterfaceId>
    <MacAddress>0212abcdefff</MacAddress>
    <IanaTechnologyType>12</IanaTechnologyType>
    <AdmissionControlSupported>No</AdmissionControlSupported>
    <PacketTaggingSupported>Yes</PacketTaggingSupported>
    <NativeQos>BestEffort</NativeQos>
    <MaxPhyRate>5000000</MaxPhyRate>
    <v2>
      <ChannelInformation>6</ChannelInformation>
    </v2>
  </Interface>
</QosDeviceCapabilities>
```

2.2.7 A_ARG_TYPE_QosDeviceState

A_ARG_TYPE_QosDeviceState is a structure that provides information about a device's current QoS state. Use of this state variable is discouraged for UPnP-QoS v3.

2.2.7.1 XML Schema Definition

This is a **string** containing an XML fragment. It contains information describing the current state of the *QosDevice* Service. The XML fragment in this argument MUST validate against

the XML schema for QosDeviceState in the XML namespace "http://www.upnp.org/schemas/QosDeviceState.xsd" which is located at "http://www.upnp.org/schemas/qos/QosDeviceState-v2.xsd".

2.2.7.2 Description of fields in the *A_ARG_TYPE_QosDeviceState* structure

QosStateId: This is a required **string** field. It MUST identify the QoS-related state of the *QosDevice* Service. In particular it MUST change after successful invocations of *SetupTrafficQos()* or *ReleaseTrafficQos()*. There may be other reasons a *QosDevice* Service changes *QosStateId*, but when the QosStateId is the same at two instances in time, all relevant *QosDevice* Service-state MUST be the same. Read the theory of operation for more details as to how this parameter is used.

Interface: This is a required structure and defines an interface. An Interface definition is required for each interface supported by the device.

2.2.7.3 Description of fields in the *Interface* structure

InterfaceId: This is a required field. Its format is defined in clause 2.2.6.2.

IpAddress: This is an optional field. This specifies the IP Address of the interface. This is optional for interfaces not configured with an IP Address. However the IP Address of configured interfaces MUST advertise this value.

InterfaceAvailability: This is a required enumeration field. The value of "0" indicates that the interface is not available. A value of "1" indicates the interface is available which may include being in power-save mode.

2.2.7.4 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<QosDeviceState
 xmlns="http://www.upnp.org/schemas/QosDeviceState.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.upnp.org/schemas/QosDeviceState.xsd
 http://www.upnp.org/schemas/qos/QosDeviceState-v2.xsd">
 <QosStateId>MyStateId001</QosStateId>
 <Interface>
   <InterfaceId>eth0</InterfaceId>
   <IpAddress>
     <Ipv4>10.10.145.24</Ipv4>
   </IpAddress>
   <InterfaceAvailability>1</InterfaceAvailability>
 </Interface>
 <Interface>
   <InterfaceId>eth1</InterfaceId>
   <InterfaceAvailability>0</InterfaceAvailability>
 </Interface>
 <Interface>
   <InterfaceId>eth2</InterfaceId>
   <IpAddress>
     <Ipv4>10.10.144.23</Ipv4>
   </IpAddress>
   <InterfaceAvailability>1</InterfaceAvailability>
 </Interface>
</QosDeviceState>
```

2.2.8 PathInformation

PathInformation is a structure that provides MAC address information about devices reachable through each active interface.

2.2.8.1 XML Schema Definition

This is a **string** containing an XML fragment. The XML fragment in this argument MUST validate against the XML schema for *PathInformation* in the XML namespace "http://www.upnp.org/schemas/PathInformation.xsd" which is located at "http://www.upnp.org/schemas/qos/PathInformation-v3.xsd".

2.2.8.2 Description of fields in *PathInformation* structure

DeviceReachableMacs: This is a required structure. *DeviceReachableMac* serves as a container for the interface specific lists of reachable MACs.

2.2.8.3 Description of fields in *DeviceReachableMacs* structure

LinkReachableMacs: This is a required structure. A *LinkReachableMacs* element is required for each available link supported by the device. For a device with physical media dedicated to an interface (such as Ethernet) there will be a *LinkReachableMacs* definition for each physical interface. For a device with a shared media (such as 802.11) there will be a *LinkReachableMacs* definition for each device pair where communication is supported by the device.

2.2.8.4 Description of fields in *LinkReachableMacs* structure

LinkId: This is a required field. Its value is of type **string** and contains the *InterfaceId* as defined in clause 2.2.6.2. Note: this field was named *LinkId* in *v2* although this field actually contains the *InterfaceId*; this name MUST be retained for backward compatibility,

BridgeId: This is an optional field. It is of type **string**. Interfaces (links) that are interconnected (bridged) within the device at L2 are identified with the same value for *BridgeId*.

MacAddress: This is an optional field. It MUST be provided when supported by the technology. It provides the MAC address of the interface for an end point device.

ReachableMac: This is an optional structure. It MUST be provided when MAC addresses are supported by the technology. It provides the MAC address(es) of end point devices that are reachable through the link, if any. The device MUST list all MAC address that it currently knows for the link.

QosSegmentId: This is a required field. Its format is defined in clause 2.2.23.2.

ActualLinkId: This is an optional field. It is of type **string**. It contains the *LinkId* as defined in clause 2.2.23.2.

2.2.8.5 Sample argument XML string – PC with two network interfaces

This is an example of an end point network device with two network interfaces.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceReachableMacs
xsi:schemaLocation="http://www.upnp.org/schemas/PathInformation.xsd
PathInformation-v3.xsd" xmlns="http://www.upnp.org/schemas/PathInformation.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <MacAddress>112233aabb03</MacAddress>
      <ReachableMac>112233aabb06</ReachableMac>
      <ReachableMac>112233aac206</ReachableMac>
      <ReachableMac>232233aa7126</ReachableMac>
      <ReachableMac>112333aab123</ReachableMac>
      <v2>
         <v3>
```

```
            <QosSegmentId>174A98172392717512321</QosSegmentId>
            <ActualLinkId>c8d7aa92c12897</ActualLinkId>
         </v3>
      </v2>
   </LinkReachableMacs>
</DeviceReachableMacs>
```

2.2.9 A_ARG_TYPE_QosDeviceInfo

A_ARG_TYPE_QosDeviceInfo is a structure that is returned to provide information concerning the specified traffic stream.

2.2.9.1 XML Schema Definition

This is a **string** containing an XML fragment. It contains transport-related information specific to the stream identified by a TrafficDescriptor, The XML fragment in this argument MUST validate against the XML schema for `QosDeviceInfo` in the XML namespace `"http://www.upnp.org/schemas/QosDeviceInfo.xsd"` which is located at `"http://www.upnp.org/schemas/qos/QosDeviceInfo-v3.xsd"`.

2.2.9.2 Description of fields in *A_ARG_TYPE_QosDeviceInfo* structure

TrafficHandle: This is a required field that identifies the Traffic Descriptor for which *QosDevice* Service information is being returned.

SourcePort: This is an optional integer field. It contains the source port of the specified traffic stream.

DestinationPort: This is an optional integer field. It contains the destination port of the specified traffic stream.

IpProtocol: This is an optional integer field. It contains the IANA assigned protocol number of the specified traffic stream.

SourceAddress: This is an optional field. It contains the source address of the specified traffic stream.

DestinationAddress: This is an optional field. It contains the destination address of the specified traffic stream.

2.2.9.3 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<QosDeviceInfo xsi:schemaLocation="http://www.upnp.org/schemas/QosDeviceInfo.xsd
QosDeviceInfo-v3.xsd" xmlns="http://www.upnp.org/schemas/QosDeviceInfo.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <TrafficHandle>TH1b4-a222-08002b34c0037f921234-723c-11b4</TrafficHandle>
   <SourcePort>554</SourcePort>
   <DestinationPort>8572</DestinationPort>
   <IpProtocol>17</IpProtocol>
</QosDeviceInfo>
```

2.2.10 A_ARG_TYPE_QosStateId

This is a **string** argument state variable. This state variable identifies the QoS-related state of the *QosDevice* Service.

2.2.11 A_ARG_TYPE_NumRotameterObservations

This is an unsigned integer state variable. This state variable indicates the number of Rotameter Observations per MAC address that a requesting Control Point is interested in receiving. If the *QosDevice* Service has this number of observations available, it MUST return

the most recent (in time) observations indicated by the number *A_ARG_TYPE_NumRotameterObservations*.

2.2.12 A_ARG_TYPE_RotameterInformation

A_ARG_TYPE_RotameterInformation is a structure that provides MAC address and Rotameter information about devices reachable through each active interface.

2.2.12.1 XML Schema Definition

This is a **string** containing an XML fragment. The XML fragment in this argument MUST validate against the XML schema for `RotameterInformation` in the XML namespace "`http://www.upnp.org/schemas/RotameterInformation.xsd`" which is located at "`http://www.upnp.org/schemas/qos/RotameterInformation-v2.xsd`".

2.2.12.2 Description of fields in A_ARG_TYPE_*RotameterInformation* structure

LinkReachableMacs: This is a required structure. A *LinkReachableMacs* definition is required for each available link supported by the device. See clause 2.2.8.4.

LinkId: This is a required field. Its value is of type **string** and contains the *InterfaceId* as defined in clause 2.2.6.2. Note: this field was named *LinkId* in v2 although this field actually contains the *InterfaceId*; this name MUST be retained for backward compatibility. It MUST be unique within the *QosDevice* Service.

QosSegmentId: This is an optional field. Its value is of type **string**. It MUST be unique within the *QosDevice* Service. It uniquely identifies a network segment of a shared media technolgy. Refer to description of *QosSegmentId* field in the clause 2.2.23.2 below for more details.

MacAddress: This is an optional field. It MUST be provided when supported by the technology. It provides the MAC address of the interface for an end point *QosDevice* Service. This is optional because not all interfaces are configured with a MAC Address. *MacAddress* may not be applicable to devices such as individual ports on switches.

BridgeId: This is an optional field. It is of type **string**. Interfaces (links) that are interconnected (bridged) within the device at L2 are identified with the same value for *BridgeId*

RotameterObservation: This is a repeating field. Each instance contains information describing a single rotameter observation. This field is defined in clause 2.2.12.3

2.2.12.3 Description of fields in the RotameterObservation *structure*

RotameterObservation structure contains the following elements that describe a Rotameter Observation

RotameterIndex: This is an integer index that is incremented and is unique per observation on the reporting *QosDevice* Service. This can serve to correlate overlapping history snapshots to determine where they overlap.

MonitorResolutionPeriod: This integer field specifies, in seconds, how often a Rotameter observation is initiated (See Figure 2-1).

ROPeriod: This integer field specifies the duration in seconds over which the *RotameterObservation* is performed. ROPeriod MUST be less than or equal to the *MonitoringResolutionPeriod*. (See Figure 2-1)

ReportingDateTime: This is a Time of Completion of Observation Period. This is wall clock time formatted per [RFC3339]; which can potentially be non-synchronized with other devices on the network.

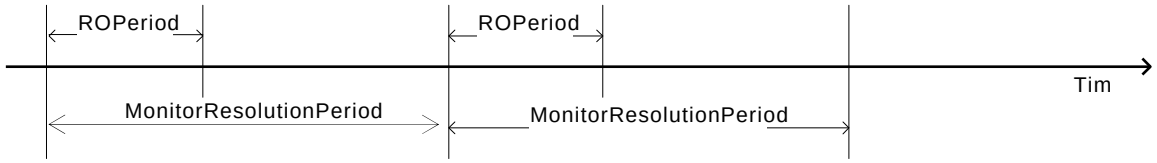


Figure 2-1 — Relationship between ROPeriod and MonitorResolutionPeriod

ROAddr: This is the MAC address of an interface on a *QosDevice* Service at which the Rotameter information is provided in the report. This parameter can be configured by the Control Point.

If it is the same address as the *MacAddress* field on the reporting *QosDevice* Service, then the Rotameter Observation is for all traffic to/from that interface on a reporting *QosDevice* Service identified by the ROAddr field.

If the address is different than the *MacAddress* field on the reporting *QosDevice* Service, then the Rotameter Observation is for all traffic between the interface identified by the *ROAddr* field on another device (which may not be a *QosDevice* Service) and the interface on a reporting *QosDevice* Service identified by the MacAddress field.

If the address is all zeros, then the report is for the entire QoS Segment of the shared media technology identified by the *QosSegmentId*. For example, this will be set to all zeros by a *QosDevice* Service on a Layer2 scheduler of a QoS Segment when reporting diagnostic information for the entire QoS Segment.

ROBits: integer field is a total number of bits of all streams (best effort, prioritized and parameterized) measured in the Observation period (ROPeriod).

ROBits0: (Optional): This integer field represents number of bits interpreted as *TrafficImportanceNumber* 0 of strictly prioritized flows in the Observation period.

ROBits1: (Optional): This integer field represents number of bits interpreted as *TrafficImportanceNumber* 1 of strictly prioritized flows in the Observation period.

ROBits2: (Optional): This integer field represents number of bits interpreted as *TrafficImportanceNumber* 2 of strictly prioritized flows in the Observation period.

ROBits3: (Optional): This integer field represents number of bits interpreted as *TrafficImportanceNumber* 3 of strictly prioritized flows in the Observation period.

ROBits4: (Optional): This integer field represents number of bits interpreted as *TrafficImportanceNumber* 4 of strictly prioritized flows in the Observation period

ROBits5: (Optional): This integer field represents number of bits interpreted as *TrafficImportanceNumber* 5 of strictly prioritized flows in the Observation period

ROBits6: (Optional): This integer field represents number of bits interpreted as *TrafficImportanceNumber* 6 of strictly prioritized flows in the Observation period

ROBits7: (Optional): This integer field represents number of bits interpreted as *TrafficImportanceNumber* 7 of strictly prioritized flows in the Observation period

It is recommended that the *QosDevice* Service employs separate priority queues for different traffic types and manages separate traffic counters (total number of bits in and out of the queue) for each of these priority queues (per attached device). For example, if a WLAN AP has four priority queues (background, best-effort, video, and voice) and is capable of managing separate counters for each of these queues, each of *ROBits1*, *ROBits0*, *ROBits5*, and *ROBits7* respectively should be implemented for each attached device. If the *QosDevice* Service is unable to manage separate counters for each priority queue (per attached device), implementing a single counter (*ROBits*) per attached device is a reasonable compromise.

To further this example, *ROPeriod* and *MonitorResolutionPeriod* are both set to 1 second and there is only a single WLAN STA attached to the WLAN AP. If the AP supports per-priority counters and the attached STA sends two bursts of traffic as follows: One at 1 Mbps for 1 second with no priority (i.e., best-effort), followed by another at 6 Mbps for 1 second with video priority (equal to 5). Then the counters for two requested observations would contain:

Observation #1: *ROBits* (1000000), and *ROBits0* (1000000). If only a single counter per-device is possible for this AP, the single counter would contain: *ROBits* (1000000).

Observation #2: *ROBits* (6000000), and *ROBits5* (6000000). If only a single counter per-device is possible for this AP, the single counter would contain: *ROBits* (6000000).

If only a single observation was requested, the most recent would be returned, i.e., Observation #2 above. If an observation was requested after 1.5 seconds, i.e., between observation periods, the most recent complete observation would be returned (#1 above).

ROBitsParameterized: This is a mandatory parameter. This is the total number of bits of all parameterized streams measured in the Observation period (*ROPeriod*).

ROPacketsParameterized: This is a mandatory parameter. This is the total number of packets of all parameterized streams measured in the Observation period (*ROPeriod*)

ROParameterizedPacketsDropped: This is a mandatory parameter. This is the total number of packets dropped for all the parameterized streams in the Observation period (*ROPeriod*).

ROPerStreamObservation: This is an optional field. Each instance of this repeating field contains information about a specific stream. This field is defined in more details in Clause 2.2.12.4

2.2.12.4 Description of fields in the ROPerStreamObservation *structure*

ROPerStreamObservation structure contains the following elements that describe a Rotameter Observations per stream.

TrafficHandle: This is an optional field of type **string**. This identifies a unique UPnP traffic stream flowing through the *QosDevice* Service. A Control Point can optionally provide a list of TrafficHandles for which it would like to obtain diagnostic information. See [QM] for the detailed definition of *TrafficHandle.* At least one of *TrafficHandle* and *Layer2StreamId* MUST be present in *ROPerStreamObservation*.

Layer2StreamId: This is an optional field. Its value is of type **string** and it is 64 characters in length. This identifies a unique traffic stream flowing through a QoS Segment identified by a *QosSegmentId*. If an L2 Technology supported by a *QosDevice* Service cannot track per stream information, then this field may be absent. A Control Point can optionally provide a specific list of *Layer2StreamIds* about which it would like to obtain diagnostic information. *Layer2StreamId* is defined in clause 2.2.17.7.

StreamBitsTransmitted: (Optional) This is the total number of bits transmitted of a parameterized traffic stream identified by a *Layer2StreamId* in the ObservationPeriod.

StreamPacketsTransmitted: (Optional) This is the total number of packets transmitted of a parameterized traffic stream identified by a *Layer2StreamId* in the ObservationPeriod.

StreamBitsReceived: (Optional) This is the total number of bits received of a parameterized traffic stream identified by a *Layer2StreamId* in the ObservationPeriod.

StreamPacketsReceived: (Optional) This is the total number of packets received of a parameterized traffic stream identified by a *Layer2StreamId* in the ObservationPeriod.

StreamPacketsDropped: (Optional) This is the total number of dropped packets of a parameterized traffic stream identified by a *Layer2StreamId* in the ObservationPeriod.

2.2.12.5 Sample argument XML string – PC with two network interfaces

This is an example of an end point network device with two network interfaces that are not currently making Rotameter Observations.

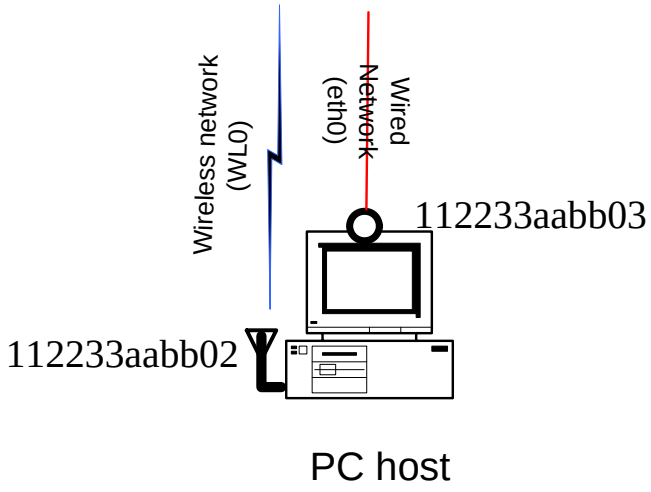


Figure 2-2 — PC with Two Network Interfaces

Example Network Rotameter Observation on a PC with two interfaces

```
<?xml version="1.0" encoding="UTF-8"?>
<RotameterInformation
xsi:schemaLocation="http://www.upnp.org/schemas/RotameterInformation.xsd
RotameterInformation-v3-20071209.xsd"
xmlns="http://www.upnp.org/schemas/RotameterInformation.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <v3>
         <QosSegmentId>1579172877A91C2DE</QosSegmentId>
         <MacAddress>112233aabb02</MacAddress>
      </v3>
   </LinkReachableMacs>
   <LinkReachableMacs>
      <LinkId>WL0</LinkId>
      <v3>
         <QosSegmentId>0069172877A91C23A</QosSegmentId>
         <MacAddress>112233aabb03</MacAddress>
      </v3>
   </LinkReachableMacs>
</RotameterInformation>
```

2.2.12.6 Sample argument XML string – PC with two network interfaces that are both end point device

Similar to the previous example this is an example of an end point network device with two network interfaces. In this example the interfaces are actively connected and actively making Rotameter Observations.

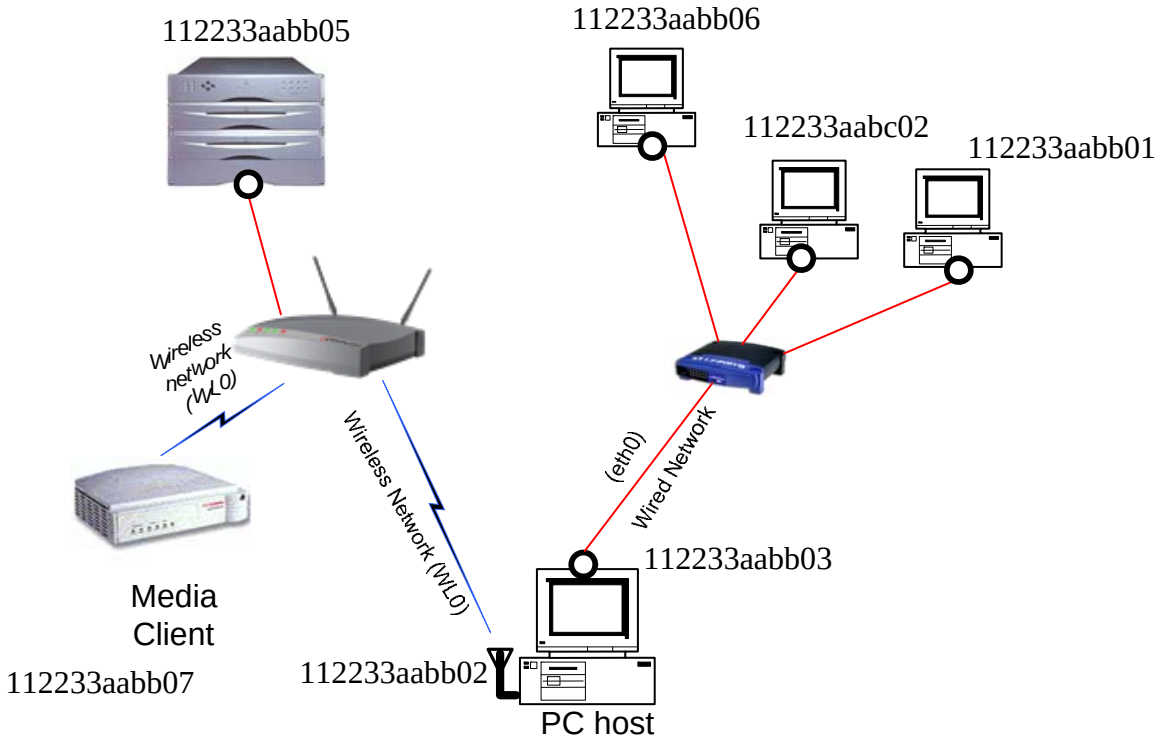


Figure 2-3 — Example of a PC connected to an active network

```
<?xml version="1.0" encoding="UTF-8"?>
<RotameterInformation
xsi:schemaLocation="http://www.upnp.org/schemas/RotameterInformation.xsd
RotameterInformation-v3-20071209.xsd"
xmlns="http://www.upnp.org/schemas/RotameterInformation.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <v3>
         <QosSegmentId>157A9172877A91C2DE</QosSegmentId>
         <MacAddress>112233aabb03</MacAddress>
      </v3>
      <RotameterObservation>
         <RotameterIndex>0</RotameterIndex>
         <ROAddr>0077c71ad71a</ROAddr>
         <ROBits>0</ROBits>
         <ROBits0>0</ROBits0>
         <ROBits1>0</ROBits1>
         <ROBits2>0</ROBits2>
         <ROBits3>0</ROBits3>
         <ROBits4>0</ROBits4>
         <ROBits5>0</ROBits5>
         <ROBits6>0</ROBits6>
         <ROBits7>0</ROBits7>
         <ROPeriod>10</ROPeriod>
         <ReportingDateTime>2006-12-19T16:39:57-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>100</MonitorResolutionPeriod>
         <v3>
            <ROBitsParameterized>0</ROBitsParameterized>
            <ROPacketsParameterized>0</ROPacketsParameterized>
            <ROParameterizedPacketsDropped>
0
```

```
</ROParameterizedPacketsDropped>
            <ROPerStreamInformation>
               <TrafficHandle>TH-782771240-777273</TrafficHandle>
               <Layer2StreamId>
                  00000000000000000000000000000000000000000000000000000382d2a1
                </Layer2StreamId>
               <StreamBitsTransmitted>0</StreamBitsTransmitted>
               <StreamPacketsTransmitted>0</StreamPacketsTransmitted>
               <StreamBitsReceived>0</StreamBitsReceived>
               <StreamPacketsReceived>0</StreamPacketsReceived>
               <StreamPacketsDropped>0</StreamPacketsDropped>
            </ROPerStreamInformation>
         </v3>
      </RotameterObservation>
   </LinkReachableMacs>
</RotameterInformation>
```

2.2.13 A_ARG_TYPE_ConfRotameterObservations

This structure is used to configure how Rotameter observations are made. Some of the configuration parameters include time period over which the rotameter observation is made, the MAC address of the device where the observation is made, specific traffic stream for which the observation is requested, etc.

2.2.13.1 XML Schema Definition

This is a **string** containing an XML fragment. It contains information specifying Rotameter configuration. The XML fragment in this argument MUST validate against the XML schema for `ConfRotameterObservations` in the XML namespace "`http://www.upnp.org/schemas/ConfRotameterObservations.xsd`" which is located at "`http://www.upnp.org/schemas/qos/ConfRotameterObservations-v2.xsd`".

2.2.13.2 Description of fields in A_ARG_TYPE_ConfRotameterObservations structure

ROPeriod: This is a mandatory field. Refer to the description of *ROPeriod* field in clause 2.2.12.3 above.

MonitorResolutionPeriod: This is a mandatory field. Refer to the description of *MonitorResolutionPeriod* field in clause 2.2.12.3 above.

PerStreamConfiguration: This optional structure contains information regarding rotameter observation configuration for each stream.

2.2.13.3 Description of fields in PerStreamConfiguration structure

ROAddr: This is an optional field. Refer to description of *ROAddr* field in clause 2.2.12.3 above.

If value of all zeros is provided, the device reports information about the traffic for the entire QoS Segment identified by *QosSegmentId*. If non-zero value is provided, the device reports information about traffic only between the device identified by the *ROAddr* field and its interface identified by the *MacAddress* field. If no value is specified, the device either reports traffic to/from itself or between itself and other devices on the network. See clause 2.3.11.2 for more details.

TrafficHandle: This is an optional field. This identifies a unique UPnP traffic stream flowing through a *QosDevice* Service. A Control Point optionally specifies this field if it is interested in obtaining information for a specific UPnP traffic stream.

QosSegmentId: This is an optional field. It uniquely identifies a network segment of a shared media technolgy. A Control Point optionally specifies this field if it is interested in obtaining information for a specific traffic stream identified by *Layer2StreamIds* on that *QosSegmentId*.

Layer2StreamId: This is an optional field. This identifies a unique traffic stream flowing through a QoS Segment identified by a *QosSegmentId*. A Control Point optionally specifies this field if it is interested in obtaining information for a specific traffic stream identified by a *Layer2StreamId* on the specified *QosSegmentId*. *Layer2StreamId* is defined in clause 2.2.17.7.

2.2.13.4 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<ConfRotameterObservations
xsi:schemaLocation="http://www.upnp.org/schemas/ConfRotameterObservations.xsd
ConfRotameterObservations-v3-20071209.xsd"
xmlns="http://www.upnp.org/schemas/ConfRotameterObservations.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <ROPeriod>10</ROPeriod>
   <MonitorResolutionPeriod>100</MonitorResolutionPeriod>
   <v2>
      <v3>
         <PerStreamConfiguration>
            <ROAddr>0077c71ad71a</ROAddr>
            <TrafficHandle>TH-000098-a787d71d</TrafficHandle>
            <QosSegmentId>174A9274F87AC71EDA1</QosSegmentId>
            <Layer2StreamId>
               00000000000000000000000000000000000000000000000000000382d2a1
            </Layer2StreamId>
         </PerStreamConfiguration>
      </v3>
   </v2>
</ConfRotameterObservations>
```

2.2.14 MostRecentStreamAction

MostRecentStreamAction is a structure that contains counters to indicate changes resulting from v2 *QosDevice* Service traffic stream actions, e.g., *SetupTrafficQos()* and *ReleaseTrafficQos()*. When the appropriate traffic stream action is successfully invoked on the *QosDevice* Service, the counter is incremented. This state variable is optional but when implemented MUST be evented to identify when QoS is setup or removed for a traffic stream. This event can be useful for diagnostic purposes, e.g. identifying which source device started or stopped a QoS-enabled traffic stream that may be contending with an ongoing stream. Further queries may be done to gain relevant information about the stream, such as querying *GetQosState()* or the *GetRotameterInformation()*, or examine the TrafficDescriptor that identifies traffic stream and policy information. This information could be displayed to an end user interested in diagnosing a streaming problem.

2.2.14.1 XML Schema Definition

This is a **string** containing an XML fragment. The XML fragment in this argument MUST validate against the XML schema for *MostRecentStreamAction* in the XML namespace "http://www.upnp.org/schemas/MostRecentStreamAction.xsd" which is located at "http://www.upnp.org/schemas/qos/MostRecentStreamAction-v2.xsd".

2.2.14.2 Description of fields in the *MostRecentStreamAction* structure

SetupTrafficQos: An unsigned integer value representing the number of successful invocations of *SetupTrafficQos()* on the *QosDevice* Service.

ReleaseTrafficQos: An unsigned integer value representing the number of successful invocations of *RemoveTrafficQos()* on the *QosDevice* Service.

2.2.14.3 Sample Argument XML String

```
<?xml version="1.0" encoding="UTF-8"?>
<MostRecentStreamAction xsi:schemaLocation="urn:schemas-upnp-
org:qos:MostRecentStreamAction MostRecentStreamAction-v2.xsd" xmlns="urn:schemas-
upnp-org:qos:MostRecentStreamAction" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <SetupTrafficQos>2</SetupTrafficQos>
   <ReleaseTrafficQos>1</ReleaseTrafficQos>
</MostRecentStreamAction>
```

2.2.15 A_ARG_TYPE_MaxPossibleRotameterObservations

A_ARG_TYPE_MaxPossibleRotameterObservations is an unsigned integer field representing the maximum number of observations that the device is capable of providing.

2.2.16 A_ARG_TYPE_Resource

A_ARG_TYPE_Resource is a required state variable. It identifies and describes the target resource in the *QosDevice* Service of a *ReleaseAdmittedQos()*, *AdmitTrafficQos()* or *UpdateAdmittedQos()* action.

2.2.16.1 XML Schema Definition

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

2.2.16.2 Description of fields in A_ARG_TYPE_Resource structure

The *A_ARG_TYPE_Resource* structure MUST contain one of the following structures:

DeviceResource: This structure identifies a resource on the device.

NetworkResource: This structure identifies and describes a specific network resource on the device.

2.2.16.3 Description of fields in DeviceResource structure

DeviceResourceId: This required field is of type **string** and MUST be one of the DeviceResourceId field values provided in the *A_ARG_TYPE_QosDeviceExtendedState* state variable.

2.2.16.4 Description of fields in NetworkResource structure

The following fields identify the specific network resource which may be an interface or a link:

InterfaceId: This is a required field of type **string**; its format is defined in clause 2.2.6.2.

QosSegmentId: This is a required field of type **string**. It is used to identify the QoS Segment on which the Interface resides. Its format is defined in clause 2.2.23.2.

LinkId: This is an optional field of type **string**. It is a unique value for a given link within the interface of the *QosDevice* Service. Its value MUST be one of the *LinkId* field values provided in the *A_ARG_TYPE_QosDeviceExtendedState* state variable.

The following fields give the *QosDevice* Service information to aid its determination of what action it should take regarding this resource (see clause 2.5.1). Please note that the indications of *QosDevice*:3 Services being upstream or downstream are from the perspective

of the current *QosDevice* Service and within the context of the stream that is the subject of the action in which the *A_ARG_TYPE_Resource* state variable is received.

QDDownstream: This is a required field. It contains a **boolean** value which = "**1**" if there is a *QosDevice*:3 Service downstream (towards the destination of the stream) within the same QoS Segment for which the *AdmitTrafficQos()* (or *UpdateAdmittedQos()*) action has been invoked.

QDUpstream: This is a required field. It contains a **boolean** value = "**1**" if there is a *QosDevice*:3 Service upstream (towards the source of the stream) within the same QoS Segment for which the *AdmitTrafficQos()* (or *UpdateAdmittedQos()*) action has been invoked.

2.2.16.5 Sample Argument XML String

```
<?xml version="1.0" encoding="UTF-8"?>
<Resource xsi:schemaLocation="urn:schemas-upnp-org:qos:Resource Resource-v3.xsd"
xmlns="urn:schemas-upnp-org:qos:Resource"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <NetworkResource>
      <InterfaceId>plc0</InterfaceId>
      <QosSegmentId>174A76932C87A219EF0C</QosSegmentId>
      <QDDownstream>1</QDDownstream>
      <QDUpstream>0</QDUpstream>
   </NetworkResource>
</Resource>
```

2.2.17 A_ARG_TYPE_AdmitTrafficQosExtendedResult

A_ARG_TYPE_AdmitTrafficQosExtendedResult is a required state variable. This state variable contains information about the status of a requested traffic stream. If admission of the traffic stream fails, this state variable provides information regarding the cause.

2.2.17.1 XML Schema Definition

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

2.2.17.2 Description of fields in A_ARG_TYPE_AdmitTrafficQosExtendedResult structure

The *A_ARG_TYPE_AdmitTrafficQosExtendedResult* structure returns the admission state information for the network or device resources. In the case of admission success, an indication of whether the *QosDevice* Service invoked the underlying Admission Mechanism is provided. In the case of admission failure, this structure returns the reason(s) for the failure. The *A_ARG_TYPE_AdmitTrafficQosExtendedResult* *structure* consists of the following fields:

TrafficHandle: This is a required field. It identifies the *TrafficDescriptor* with which the admission status is associated.

Exactly one of the following structures MUST be provided:

AdmissionStatusNet: Required when an admission request is made for network resources. Defined in clause 2.2.17.3.

AdmissionStatusDev: Required when an admission request is made for device resources. Defined in clause 2.2.17.5.

2.2.17.3 Description of fields in the AdmissionStatusNet structure

The AdmissionStatusNet is a structure that consists of a list of entries which provides the admission status of an instance of a traffic descriptor within the QoS segment. Here are the details of the parameters:

Reason: This field is a **string** that contains either a reason code (see Table 2-1) or a text string identifying the reason why the requested action was not accomplished. The *Reason* field is a required field; it MAY appear more than once. There are two types of reasons: “standard” and “non-standard”:

- Standard reasons are abbreviated as a three digit number, as shown in Table 2-2.
- Non-standard reasons are identified by a text string to describe the reason. Non-standard reasons can be used for testing, plugfests, vendor specific reasons, etc.

ListOfLayer2StreamIds: This optional structure contains one or more tuples identifying the streams that may be blocking the stream whose admission has failed on this QoS Segment. It may or may not be possible for the *QosDevice* Service to include all Layer2StreamIds on this QoS Segment. It is defined in clause 2.2.17.7.

AllocatedResources: This is an optional field that indicates the resources that have been reserved for a successful admit or update request. The format of this field is defined in clause 2.2.17.4.

2.2.17.4 Description of fields in the AllocatedResources structure

The *AllocatedResources* is a structure that contains information that indicates the resources that have been allocated for a successful admit or update request.

MaxCommittedDelay: This optional field is of type ui4. This is the maximum of the QoS Segment Delay that this QosDevice has committed to provide on this QoS Segment. See [QD_Add] for technology-specific details.

MaxCommittedJitter: This optional field is of type ui4. This is the maximum of the QoS Segment Jitter that this QosDevice has committed to provide on this QoS Segment. See [QD_Add] for technology-specific details.

Table 2-2 — Reason Codes For AdmissionStatusNet

ReasonCode	Reason Description	Description
000	Success	Traffic stream was admitted as requested.
001	Registered	This *QosDevice* Service has determined that another *QosDevice* Service on the QoS Segment is responsible for the request. It is remaining passive but has registered the Traffic stream and is aware of it.
762	Insufficient resources	Action failed due to insufficient resources.
764	Admission Control Not Supported	Admission Control is not supported on this QoS Segment.

2.2.17.5 Description of fields in the AdmissionStatusDev structure

DeviceId: This is a required field. This field is a **string** that uniquely identifies a particular Device Resource within the *QosDevice* Service. The *DeviceId* provided in this field MUST exist in the device as reported in the *QosDeviceExtendedState* state variable.

Reason: This field is a **string** that contains either a reason code or a text string identifying the reason why the requested action was not accomplished. The *Reason* field is a required field; it MAY appear more than once. There are two types of reasons: “standard” and “non-standard”:

Standard reasons are abbreviated as a three digit number, as shown in Table 2-3.

Non-standard reasons are identified by a text string to describe the reason. Non-standard reasons can be used for testing, plugfests, vendor specific reasons, etc.

Table 2-3 — Reason Codes For AdmissionStatusDev

ReasonCode	ReasonDescription	Description
000	Success	Traffic stream was admitted as requested.
762	Insufficient resources	Action failed due to insufficient resources.
764	Admission Control Not Supported	Admission Control is not supported on this Device Resource.

ListOfLayer2StreamIds: This optional structure contains one or more tuples identifying the admitted streams that may be blocking the stream whose admission has failed on this Device Resource (no prioritized streams will be included since they cannot possibly be blocking). It is defined in clause 2.2.17.6.

2.2.17.6 Description of fields in the ListOfLayer2StreamIds structure

NumberOfLayer2StreamIds: This is a required unsigned integer field. Its value is the number of entries in *ListOfLayer2StreamIds*.

Layer2Stream: This is an optional repeating structure (clause 2.2.17.7), each instance identifying a particular stream on the device. The number of *Layer2Stream* structures present MUST equal *NumberOfLayer2StreamIds*.

2.2.17.7 Description of fields in the Layer2Stream structure

Layer2StreamId: This is a required field of type **string**. It consists of 64 characters which provide an L2 Stream ID that uniquely identifies a particular stream to all *QosDevice* Services on the QoS Segment. The interpretation of this field is dependent upon the L2 Technology, several of which are described in the UPnP QosDevice:3 Underlying Technology Interface Addendum [QD_Add] . The QosManager treats this as a string, using it for matching purposes when looking for the *TrafficDescriptor* associated with the blocking stream. A given stream may have a different *Layer2StreamId* on each QoS Segment it traverses. Details on the formation of a Layer2StreamId based on the underlying technology can be found in the reference document [QD_Add]

TrafficDescriptorAvailable: This is a required field of type **boolean**: "**1**" indicates that the *QosDevice* Service can provide the *TrafficDesciptor* for the Blocking Stream if requested; "0" indicates that it can not.

2.2.17.8 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<AdmitTrafficQosExtendedResult xsi:schemaLocation="urn:schemas-upnp-
org:qos:AdmitTrafficQosExtendedResult AdmitTrafficQosExtendedResult-v3.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn:schemas-upnp-
org:qos:AdmitTrafficQosExtendedResult">
   <TrafficHandle>TH01234567</TrafficHandle>
   <AdmissionStatusNet>
      <Reason>000</Reason>
      <AllocatedResources>
         <MaxCommitedDelay>2000</MaxCommittedDelay>
         <MaxCommittedJitter>500</MaxCommittedJitter>
      </AllocatedResources>
   </AdmissionStatusNet>
</AdmitTrafficQosExtendedResult>
```

2.2.18 A_ARG_TYPE_ListOfAdmittedTraffic

This is a required state variable. It contains the list of admitted traffic streams and associated resources known to this *QosDevice* Service.

2.2.18.1 XML Schema Definition

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

2.2.18.2 Description of fields in the ListOfAdmittedTraffic structure

The *ListOfAdmittedTraffic* is a required structure containing zero or more entries of *AdmittedTrafficItem* and a count of the number of *AdmittedTrafficItem* structures provided.

NumberOfAdmittedTrafficItems: This is a required unsigned integer field. It contains the number of *AdmittedTrafficItem* instances in the list.

AdmittedTrafficItem: This is an optional structure. There MUST be one instance of this structure for each stream on the device. The number of *AdmittedTrafficItem* structures present MUST equal *NumberOfAdmittedTrafficItems*.

2.2.18.3 Description of fields in the AdmittedTrafficItem structure

Each *AdmittedTrafficItem* lists one *TrafficDescriptor*, followed by several elements providing information about the stream: The scope of the TrafficDescriptors can range from a specific set of Traffic Descriptors to every Traffic Descriptor registered on the device.

TrafficDescriptor: This required field contains a TrafficDescriptor currently active on the *QosDevice* Service.

NetworkResource: Either this structure or *DeviceResource*, but not both, MUST be present. It contains information about the network resources for which QoS was setup for this *AdmittedTrafficItem*.

DeviceResource: Either this structure or *NetworkResource*, but not both, MUST be present. It contains information about the device resources for which QoS was setup for this *AdmittedTrafficItem*.

2.2.18.3.1 Description of the fields in the NetworkResource structure

InterfaceId: This is a required field of type **string**; its format is defined in clause 2.2.6.2. This field contains the *InterfaceId* associated with this *AdmittedTrafficItem*.

QosSegmentId: This is a required field that uniquely identifies a network segment. This is a **string** that is used by the QosManager to determine the *QosDevice* Services that are attached to a specific segment. Its format is defined in clause 2.2.23.2.

LinkId: This is an optional field of type **string**; its format is defined in clause 2.2.23.2. If present, this field contains the *LinkId* associated with this *AdmittedTrafficItem*.

Layer2StreamId: This is an optional field; it is defined in clause 2.2.17.7. If present, this field contains the *Layer2StreamId* associated with this *AdmittedTrafficItem*.

AdmissionStatusNet: This optional structure contains information about the admission status associated with this *AdmittedTrafficItem*. See clause 2.2.17.3.

2.2.18.3.2 Description of the fields in DeviceResource structure

DeviceResourceId: This is an optional field of type **string**; its format is defined in clause 2.2.16.3. If present, this field contains the *DeviceResourceId* associated with this *AdmittedTrafficItem*.

AdmissionStatusDev: This optional structure contains information about the admission status associated with this *AdmittedTrafficItem*. See clause 2.2.17.5.

2.2.18.4 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<ListOfAdmittedTraffic xsi:schemaLocation="urn:schemas-upnp-
org:qos:ListOfAdmittedTraffic ListOfAdmittedTraffic-v3.xsd" xmlns="urn:schemas-
upnp-org:qos:ListOfAdmittedTraffic"
xmlns:td="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:at="urn:schemas-upnp-
org:qos:AdmitTrafficQosExtendedResult">
   <NumberOfAdmittedTrafficItems>1</NumberOfAdmittedTrafficItems>
   <AdmittedTrafficItem>
      <TrafficDescriptor>
         <td:TrafficHandle>TH1b4-a222-08002b34c0037f921234-723c-
11b4</td:TrafficHandle>
         <td: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>
         </td:TrafficId>
         <td:AvailableOrderedTspecList>
            <td:Tspec>
               <td:TspecIndex>1</td: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>
            </td:Tspec>
         </td: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>
            <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>
      <NetworkResource>
         <InterfaceId>eth0</InterfaceId>
         <QosSegmentId>006957C7AFF8E2A0D</QosSegmentId>
         <Layer2StreamId>
00000000000000000000000000000000000000000000000000000382d2a1
         </Layer2StreamId>
         <AdmissionStatusNet>
            <at:Reason>000</at:Reason>
         </AdmissionStatusNet>
      </NetworkResource>
   </AdmittedTrafficItem>
</ListOfAdmittedTraffic>
```

2.2.19 A_ARG_TYPE_PreferredQph

This is an optional state variable. It contains information pertaining to a *QosPolicyHolder* Service that is either preferred or a candidate for preference. For information on *QosPolicyHolder* Services and their preference see [QPH].

2.2.19.1 XML Schema Definition

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

2.2.19.2 Description of fields in the PreferredQph structure

PreferredQphId: This **string** is the ServiceId of the Preferred *QosPolicyHolder* Service [refer to QPH].

QphPreferenceCount: This field is an integer which is incremented by the *QosPolicyHolder* Service each time a new Preferred *QosPolicyHolder* Service is identified. It is used to resolve conflicts between *QosPolicyHolder* Services regarding which is preferred. The logic for this is described in [QPH].

2.2.19.3 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<PreferredQph xsi:schemaLocation="urn:schemas-upnp-org:qos:PreferredQph
PreferredQph-v3.xsd" xmlns="urn:schemas-upnp-org:qos:PreferredQph"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <PreferredQphId>
2fac1234-31f8-11b4-a222-08002b34c003:urn:upnp-org:serviceId:QosPolicyHolder-3a
   </PreferredQphId>
   <QphPreferenceCount>7</QphPreferenceCount>
</PreferredQph>
```

2.2.20 UnexpectedStreamChange

UnexpectedStreamChange is an optional unsigned integer which is incremented by one for each change to a stream's reservation that is detected by the *QosDevice* Service. This state variable MUST be evented if implemented. It MUST be implemented if the *GetUnexpectedStreamChanges()* action is implemented.

Examples of unexpected stream changes are: a traffic stream has changed because of preemption of the stream by a *QosManager* (released or downgraded to use fewer resources) or because of some spontaneous L2 occurrence (release, downgrade, upgrade or failure). The details of how the *QosDevice* Service handles and L2 detects these occurences are implementation-specific.

2.2.21 A_ARG_TYPE_PreemptingTrafficInfo

This is a required state variable. This state variable identifies the traffic stream that caused the invocation of *UpdateAdmittedQos()* or *ReleaseAdmittedQos()* action as a result of preemption (as indicated by *ReleaseCausedByPreemption* having a value = "**1**").

2.2.21.1 XML Schema Definition

This is a **string** containing an XML fragment. It identifies the stream (if any) which preempted this traffic stream. The XML fragment in this argument MUST validate against the XML schema for `PreemptingTrafficInfo` in the XML namespace "`urn:schemas-upnp-org:qos:PreemptingTrafficInfo`" which is located at "`http://www.upnp.org/schemas/qos/PreemptingTrafficInfo-v3.xsd`".

2.2.21.2 Description of fields in the PreemptingTrafficInfo structure

ReleaseCausedByPreemption: This is a required field of type **boolean**. A value of "**1**" is used if the release (or downgrade) is caused by preemption, otherwise a value of "**0**" is used.

PreemptingTrafficDescriptor: This optional field identifies the Traffic Descriptor of the preempting stream (if any).

2.2.21.3 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<PreemptingTrafficInfo xsi:schemaLocation="urn:schemas-upnp-
org:qos:PreemptingTrafficInfo PreemptingTrafficInfo-v3.xsd" xmlns="urn:schemas-
upnp-org:qos:PreemptingTrafficInfo"
xmlns:td="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <ReleaseCausedByPreemption>true</ReleaseCausedByPreemption>
   <PreemptingTrafficDescriptor>
```

```
      <td:TrafficHandle>TH1b4-a222-08002b34c0037f921234-723c-
11b4</td:TrafficHandle>
      <td: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>
      </td:TrafficId>
      <td:AvailableOrderedTspecList>
         <td:Tspec>
            <td:TspecIndex>1</td: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>
         </td:Tspec>
      </td: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>
         <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>
   </PreemptingTrafficDescriptor>
</PreemptingTrafficInfo>
```

2.2.22 A_ARG_TYPE_ListOfMostRecentUnexpectedStreamChanges

The *A_ARG_TYPE_ListOfMostRecentUnexpectedStreamChanges* state variable is optional. It contains a list of the most recent unexpected stream changes (preemptions or spontaneous L2-caused changes) that have occurred on the *QosDevice* Service.

2.2.22.1 XML Schema Definition

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

2.2.22.2 Description of fields in the ListOfMostRecentUnexpectedStreamChanges *structure*

ListOfMostRecentUnexpectedStreamChanges is an optional structure. It contains zero or more entries, each of which describes an unexpected stream change. It also contains an integer value of the number of *UnexpectedStreamChange* structures that are present in the list.

NumberOfRecentUnexpectedStreamChanges: This is a required integer variable which specifies how many instances (zero or more) of *UnexpectedStreamChange* are present in the list.

UnexpectedStreamChange: This structure contains information describing a single unexpected stream change. This structure is defined in clause 2.2.22.3.

2.2.22.3 Description of fields in the UnexpectedStreamChange structure

UnexpectedStreamChange is a structure that contains the traffic descriptors of both the affected and the affecting streams and an indication of whether the change was a release or an update. The number of instances of *UnexpectedStreamChange* present MUST equal *NumberOfRecentUnexpectedStreamChanges*.

AffectedStreamTrafficDescriptor: This is a required field. It contains the traffic descriptor of the stream that was affected at the time it was downgraded or released.

AffectingStreamTrafficDescriptor: This is an optional field. It contains the traffic descriptor of the preempting stream, if the change was caused by a UPnP action. Otherwise, this structure should not be included.

StreamReleased: This is a required field. It contains a **boolean** value which = "**1**" if the affected stream was released and "**0**" if the affected stream was updated but not released.

UnexpectedStreamChangeIndex: This is a required field that indicates the value of the *UnexpectedStreamChange* state variable at the time this particular stream change occurred. *UnexpectedStreamChange* is incremented prior to populating this field.

2.2.22.4 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<ListOfMostRecentUnexpectedStreamChanges xsi:schemaLocation="urn:schemas-upnp-
org:qos:ListOfMostRecentUnexpectedStreamChanges
ListOfMostRecentUnexpectedStreamChanges-v3.xsd" xmlns="urn:schemas-upnp-
org:qos:ListOfMostRecentUnexpectedStreamChanges"
xmlns:td="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <NumberOfRecentUnexpectedStreamChanges>1</NumberOfRecentUnexpectedStreamChanges>
   <UnexpectedStreamChange>
      <AffectedStreamTrafficDescriptor>
         <td:TrafficHandle>TH1b4-a222-08002b34c0037f921234-723c-
11b4</td:TrafficHandle>
         <td:TrafficId>
            <SourceAddress>
               <Ipv4>192.168.1.1</Ipv4>
```

```
            </SourceAddress>
            <SourcePort>554</SourcePort>
            <DestinationAddress>
               <Ipv4>192.168.1.7</Ipv4>
            </DestinationAddress>
            <DestinationPort>7534</DestinationPort>
            <IpProtocol>17</IpProtocol>
            <v2TrafficId>
               <v3TrafficId>
                  <SourceUuid>2fac1234-31f8-11b4-a324-08002b34c003</SourceUuid>
                  <DestinationUuid>7f921234-723c-11b4-a732-
2fac2b34c003</DestinationUuid>
               </v3TrafficId>
            </v2TrafficId>
         </td:TrafficId>
         <td:AvailableOrderedTspecList>
            <td:Tspec>
               <td:TspecIndex>1</td:TspecIndex>

<AvTransportUri>rtsp://192.168.1.1/Movies/Sample12.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>
            </td:Tspec>
         </td: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>
            <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>
      </AffectedStreamTrafficDescriptor>
      <AffectingStreamTrafficDescriptor>
         <td:TrafficHandle>TH1b4-a222-0037f908002b34c21234-723c-
11b4</td:TrafficHandle>
         <td: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>
         </td:TrafficId>
         <td:AvailableOrderedTspecList>
            <td:Tspec>
               <td:TspecIndex>1</td: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>
            </td:Tspec>
         </td: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>
            <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>
      </AffectingStreamTrafficDescriptor>
      <StreamReleased>1</StreamReleased>
      <UnexpectedStreamChangeIndex>1</UnexpectedStreamChangeIndex>
   </UnexpectedStreamChange>
</ListOfMostRecentUnexpectedStreamChanges>
```

2.2.23 A_ARG_TYPE_QosDeviceExtendedState

The *A_ARG_TYPE_QosDeviceExtendedState* structure contains the state information describing the *QosDevice* Service at the time the **GetQosExtendedState** action is invoked. As the name implies, this state variable is an extension of the *A_ARG_TYPE_QosDeviceState* state variable.

2.2.23.1 XML Schema Definition

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

2.2.23.2 Description of fields and organization in the QosDeviceExtendedState structure

The QoS related properties of the *QosDevice* Service are expressed in a hierarchical form. The hierarchy is represented by containers (see Figure 2-7 in clause 2.5.1). Each container defines the context of the contained properties. There are multiple container types; these are nested to match the characteristics of the device and network that they represent.

The following container types occur in *A_ARG_TYPE_QosDeviceExtendedState*:

DeviceResource: This is an optional container. This container describes the capabilities of the device which are not related to a specific interface. Examples include buffer space that is shared by multiple interfaces, transcoding resources, etc.

Interface: This is a required container that occurs one or more times. This container describes a network interface on the *QosDevice* Service. An Interface container is required for each interface supported by the device. There MUST be at least one Interface container. This information may be provided even if the physical interface is down at a given time. Note that on some devices, it may be impossible for the *QosDevice* Service to even detect the existence of an inactive interface. In this case, the Interface container for the inactive Interface will be absent—even if it has existed in the past. However, there will always be at least one available interface otherwise this *QosDevice* Service would not be accessible.

Link: This is an optional container. It contains properties that are specific to a particular link. If present, this container MUST be instantiated within an Interface container.

The properties defined by the *QosDeviceExtendedState* state variable are identified below. Table 2-4 specifies the container in which each property may appear. If a property appears in more than one container for a *QosDevice* Service, the property closest (above) in the container hierarchy is the one that applies. For example if MaxPhyRate appears in both the Interface and the Link container, the value of MaxPhyRate in the link container is specific for that link, but the value in the Interface container holds for the entire Interface and could be different.

Table 2-4 — Containers In Which A Parameter Can Appear

Name	Device	Interface	Link
DeviceResourceId	Yes	No	No
AdmitCntrlDev	Yes	No	No
InterfaceId	No	Yes	No
IanaTechnologyType	No	Yes	No
InterfaceAvailability	No	Yes	No
NativeQos	No	Yes	No
AdmitCntrlNet	No	Yes	No
PacketTaggingSupported	No	Yes	No
QosSegmentId	No	Yes	No
QosSegmentName	No	Yes	No
LinkId	No	No	Yes
PeerLinkMacAdd	No	No	Yes
IpAddress	No	Yes	Yes
MacAddress	No	Yes	Yes
MaxPhyRate	No	Yes	Yes
ChannelInformation	No	Yes	Yes
ListOfProtoTspecs	No	Yes	Yes

DeviceResourceId: This is an optional field of type **string**; its format is defined in clause 2.2.16.3.

AdmitCntrlDev: This is an optional field. It indicates if the device supports admission control to manage the device's non-interface resources (e.g. tuner, disk space, time-shift-buffer, etc.). This is a **boolean** field where "**1**" = Supported and "**0**" = NotSupported.

InterfaceId: This is a required field of type **string**; its format is defined in clause 2.2.6.2.

IanaTechnologyType: This is an optional field. The *IanaTechnologyType* (IANA uses the designation IANAifType) is an integer assigned by IANA for any media type, such as a value of 6 for 802.3 media type or a value of 71 for 802.11 media type. The allowed integer values for this parameter are specified in [IANA].

InterfaceAvailability: This is a required field of type **boolean** that MUST be provided in the *Interface* container. The value of "**0**" indicates that the interface is not available. A value of "**1**" indicates the interface is available which may include being in power-save mode. (E.g., the wireless station (STA) in power-save mode while associated with an Access Point (AP))

AdmitCntrlNet: This is an optional field of type **boolean**. If *InterfaceAvailability* field for a network interface is equal to "**1**", this field MUST be present. If the network interface supports Admission Technology, this field is set to "**1**", otherwise it is set to "**0**".

NativeQos: This is an optional field of type **string**. If *AdmitCntrlNet* field is equal to "**1**", then the *NativeQos* field MUST contain one of the following values: "Prioritized", "BestEffort", "Scheduled", "BothPrioritizedAndScheduled". If AdmitCntrlNet field is equal to "**0**", then the *NativeQos* field MUST contain one of the following values: "Prioritized", "BestEffort".

PacketTaggingSupported: This is a required field of type **boolean**. *PacketTaggingSupported* field indicates whether the device is capable of tagging L2 priorities on the outgoing interface. Field values are "**1**"= Supported and "**0**" = NotSupported

QosSegmentId: This is a mandatory field that uniquely identifies a network segment. This is a **string** that is used by the *QosManager* to determine the QosSegment to which the *QosDevice* Services is attached. It is used to identify multiple QoS Segments in the UPnP-QoS network. Refer to document [QD_Add] for details on the formation of the *QosSegmentId*.

The requirements for a *QosSegmentId* are that it MUST be unique (within the network)—i.e., not used to identify any other QosSegment)—and that it MUST be generated using a known algorithm that allows any *QosDevice* Service on the QosSegment to generate it independently. The precise details of the format for several technologies can be found in [QD_Add] . New technologies will require the definition of analogous formats to guarantee that each *QosDevice* Service on a given QosSegment can independently compute the same *QosSegmentId* that is unique for the segment.

In general, the *QosSegmentId* is formed from the concatenation of the *IANATechnologyType* of the underlying L2 network and some information that is specific to a particular instance of that L2 Technology that will guarantee the uniqueness of the *QosSegmentId*. Examples are the MAC address of the AP in an 802.11 network and the NID (Network ID) in a HomePlug AV network.

If the L2 Technology underlying a QosSegment does not have a single *IANATechnologyType*—e.g., a Layer2 QoS Bridging technology that bridges different L2 Technologies—it MUST still ensure that it generates a unique *QosSegmentId*.

QosSegmentName: This field is a **string** that provides a user friendly name for the network segment. Different *QosDevice* Services within same QosSegment MAY provide different names for a QosSegment. This is an optional field.

LinkId: This is a required field if the Link container is provided. The value is of type string and is a unique value for a given device and is used to identify the link. For a given link the values used MUST be the same as the *LinkId* field in the *PathInformation* state variable.

PeerLinkMacAdd: This is an optional field which should be provided (if known) if the Link container is provided. It provides the MAC address of the peer device on the link.

IpAddress: This is an optional field. It specifies the IP Address of the interface. This is optional because not all interfaces are configured with an IP Address. If the interface is configured with an IP Address it MUST advertise this value.

MacAddress: This is an optional field. It specifies the MAC Address of the interface. This is optional because not all interfaces are configured with a MAC Address. If the interface is configured with a MAC Address it MUST advertise this value.

MaxPhyRate: This parameter is optional and indicates the maximum PHY rate of the interface in units of bits/sec expressed as a value of type unsigned integer. If the interface is active this value indicates the maximum operating PHY rate, this is dependent on the device that it is connected to (e.g., for an Ethernet interface that supports 100/10 Mbps connected to a switch that supports 10Mbps, the value returned is 10 Mbps)

ChannelInformation: Indicates the channel number of the IanaTechnologyType, if the technology supports channels. For example, 802.11 (value=71) supports multiple channels. Expressed as a value of type unsignedInt.

ListOfProtoTspecs: This optional structure is a list of ProtoTspecs reported by the *QosDevice* Service. Each element on the list is a ProtoTspec variable as defined in clause 2.2.23.4.

2.2.23.3 Description of fields in the ListOfProtoTspecs structure

NumProtoTspecs: This field is an unsigned integer containing the number of ProtoTspecs reported by the *QosDevice* Service.

ProtoTspec: This is a repeating field, each instance is a *ProtoTspec* state variable.

2.2.23.4 ProtoTspec

This is a required field that contains a prototypical TSPEC identifying parameter values that can be supported by the *QosDevice* Service based on TrafficDescriptor provided as input and the L2 Technology. For some parameters, it may specify a range of values which can be supported. Note that it is not a TSPEC; it is intended to give information to the QosManager that assists in formation of a TSPEC that this *QosDevice* Service can support.

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

TspecIndex: This is an optional integer field. If present, it identifies the particular TSPEC within the TrafficDescriptor to which this ProtoTspec pertains. If not present, then the ProtoTspec reflects the capabilities of the underlying L2 Technology without regard to any specific TSPEC.

NumberOfParameters: This is a required integer field. It specifies how many Parameter structures are present in the ProtoTspec. Since one or more Parameter fields is expected, this field MUST have a value of 1 or greater.

Parameter: This is a required structure. It contains a set of fields which define the parameter. It may appear one or more times within the *ProtoTspec*. The number of Parameter structures present MUST equal NumberOfParameters.

2.2.23.5 Description of fields in the Parameter structure

ParameterName: This is a required field. It contains the name of the parameter. It MUST be a valid parameter name (from the TSPEC parameters defined in the QosManager [QM]).

ParameterInclusion: This is a required field that specifies whether the parameter is "Mandatory", "Recommended", "Optional", "Fixed" or "Ignored" in this particular L2 Technology and *QosDevice* Service.

ParameterMinValue: This is an optional field which specifies the minimum value of the parameter.

ParameterMaxValue: This is an optional field which specifies the maximum value of the parameter. If it contains the same value as ParameterMinValue then the parameter can have only one value.

DefaultValue: This is an optional field. It contains the default value that will be used for this parameter if the parameter is not specified in the TSPEC in the TrafficDescriptor provide by the QosManager.

2.2.23.6 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<QosDeviceExtendedState xsi:schemaLocation="urn:schemas-upnp-org:qos:QosDeviceExtendedState
QosDeviceExtendedState-v3.xsd" xmlns="urn:schemas-upnp-org:qos:QosDeviceExtendedState"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:pt="urn:schemas-upnp-
org:qos:ProtoTspec">
   <DeviceResource>
      <DeviceResourceId>String</DeviceResourceId>
   </DeviceResource>
   <Interface>
      <InterfaceId>eth0</InterfaceId>
      <IanaTechnologyType>174</IanaTechnologyType>
      <InterfaceAvailability>1</InterfaceAvailability>
      <IpAddress>
         <Ipv4>192.168.1.1</Ipv4>
      </IpAddress>
      <MacAddress>123400000054</MacAddress>
      <Link>
         <LinkId>LK7727501234</LinkId>
         <PeerLinkMacAdd>12340c8d0054</PeerLinkMacAdd>
         <MaxPhyRate>3000000</MaxPhyRate>
         <ListOfProtoTspecs>
            <NumberOfProtoTspecs>1</NumberOfProtoTspecs>
            <ProtoTspec>
               <pt:TspecIndex>1</pt:TspecIndex>
               <pt:NumberOfParameters>1</pt:NumberOfParameters>
               <pt:Parameter>
                  <pt:ParameterName>DataRate</pt:ParameterName>
                  <pt:ParameterInclusion>Mandatory</pt:ParameterInclusion>
                  <pt:ParameterMinValue>0</pt:ParameterMinValue>
                  <pt:ParameterMaxValue>30000000</pt:ParameterMaxValue>
                  <pt:DefaultValue>0</pt:DefaultValue>
               </pt:Parameter>
            </ProtoTspec>
         </ListOfProtoTspecs>
      </Link>
      <AdmitCntrlNet>1</AdmitCntrlNet>
      <PacketTaggingSupported>1</PacketTaggingSupported>
      <NativeQos>Prioritized</NativeQos>
      <MaxPhyRate>30000000</MaxPhyRate>
      <QosSegmentId>174A9818273771CD91</QosSegmentId>
      <QosSegmentName>PLC 91</QosSegmentName>
   </Interface>
</QosDeviceExtendedState>
```

2.2.24 A_ARG_TYPE_Layer2Mapping

A_ARG_TYPE_Layer2Mapping is a required state variable that is an output parameter of the *AdmitTrafficQos()* and *UpdateAdmittedQos()* actions. It reports the Layer2StreamId of the admitted/updated stream to the QosManager. The QosManager uses this information if it invokes the *SetL2Map()* action to inform those *QosDevice* Services, on the segment, that did not report the Layer2StreamId value.

2.2.24.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument MUST validate against the XML schema for `Layer2Mapping` in the XML namespace "`urn:schemas-upnp-org:qos:Layer2Mapping`" which is located at "`http://www.upnp.org/schemas/qos/Layer2Mapping-v3.xsd`".

2.2.24.2 Description of fields in the A_ARG_TYPE_Layer2Mapping state variable

This structure provides the Layer2StreamId for the admitted/updated stream on the QosSegment

TrafficHandle: This is a required field which identifies the Layer 3 stream associated with the Layer2StreamId.

QosSegmentId: This is a required field which identifies the QosSegment in which the Layer2StreamId is defined. Its format is defined in clause 2.2.23.2.

Layer2StreamId: This is a required field which provides the L2 Stream Id being mapped. Its format is defined in clause 2.2.17.7.

2.2.24.3 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<Layer2Mapping xsi:schemaLocation="urn:schemas-upnp-org:qos:Layer2Mapping
Layer2Mapping-v3.xsd" xmlns="urn:schemas-upnp-org:qos:Layer2Mapping"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <TrafficHandle>TH0ac4f92a23a</TrafficHandle>
   <QosSegmentId>174A0AC77D0A872C3A</QosSegmentId>
   <Layer2StreamId>
00000000000000000000000000000000000000000000000000000382d2a1
   </Layer2StreamId>
</Layer2Mapping>
```

2.2.25 A_ARG_TYPE_AdmitTrafficQosSucceeded

A_ARG_TYPE_AdmitTrafficQosSucceeded is a required state variable of type **boolean** that declares whether the requested action succeeded (="1") or failed (="0"). If this state variable indicates failure, other output arguments provide details of cause of the failure..

2.2.26 _A_ARG_TYPE_TrafficDescriptorsWanted

A_ARG_TYPE_TrafficDescriptorsWanted is a required state variable of type **boolean** that declares whether the calling Control Point wants (="1") or doesn't want (="0") the *QosDevice* Service to return a list of traffic descriptors.

2.2.27 A_ARG_TYPE_SetPreferredQphResults

A_ARG_TYPE_SetPreferredQphResults is an optional state variable. It is an unsigned integer that reports the result of the *SetPreferredQph()* action. The result will be one of the following reason codes expressed as an integer.

Table 2-5 — Reason Codes For *A_ARG_TYPE_SetPreferredQphResults*

ReasonCode	ReasonDescription	Description
000	Success	Preferred QosPolicyHolder Service Set Successfully
770	PreferredQph Failure	The requested (input) PreferredQph cannot be set as the preferred QosPolicyHolder Service because another QosPolicyHolder Service is preferred (with higher PreferredQphCount)
771	PrefferedQph Sync Error	A synchronization error has occurred. (the PreferredQphCount is the one currently used but it is associated with a different PreferredQphId)

2.2.28 A_ARG_TYPE_ NumberOfUnexpectedStreamChangesRequested

A_ARG_TYPE_ NumberOfUnexpectedStreamChangesRequested is an optional state variable. It is an unsigned integer that specifies the maximum number of unexpected stream changes that the *QosDevice* Service SHOULD return.

2.2.29 A_ARG_TYPE_ NumberOfUnexpectedStreamChangesReported

A_ARG_TYPE_NumberOfUnexpectedStreamChangesReported is an optional state variable. It is an unsigned integer that specifies the number of unexpected stream changes that the *QosDevice* Service is reporting. This number MUST be less than or equal to *A_ARG_TYPE_NumberOfUnexpectedStreamChangesRequested*.

2.2.30 A_ARG_TYPE_NewTrafficLeaseTime

A_ARG_TYPE_NewTrafficLeaseTime is a required state variable. It is an unsigned integer that specifies a new lease time for the identified stream. The new lease time is specified in units of milliseconds.

2.2.31 A_ARG_TYPE_TrafficDescriptorContainer

A_ARG_TYPE_TrafficDescriptorContainer is a required state variable. This is a structure which contains zero or one TrafficDescriptors. This structure allows for an optional TrafficDescriptor to be passed as an input or output argument of an action. It is an input parameter for *GetExtendedQosState()* and an output parameter for *VerifyTrafficHandle()*.

2.2.31.1 XML Schema Definition

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

2.2.31.2 Description of fields in the A_ARG_TYPE_TrafficDescriptorContainer state variable

TrafficDescriptor: This optional field contains a *TrafficDescriptor*. There MUST be at most one *TrafficDescriptor* in an *A_ARG_TYPE_TrafficDescriptorContainer.* Refer to 2.2.2 for *TrafficDescriptor* details.

2.2.31.3 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<TrafficDescriptorContainer xsi:schemaLocation="urn:schemas-upnp-
org:qos:TrafficDescriptorContainer TrafficDescriptorContainer-v3.xsd"
xmlns="urn:schemas-upnp-org:qos:TrafficDescriptorContainer"
xmlns:td="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <TrafficDescriptor>
      <td:TrafficHandle>TH1b4-a222-08002b34c0037f921234-723c-
11b4</td:TrafficHandle>
      <td: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>
      </td:TrafficId>
      <td:AvailableOrderedTspecList>
         <td:Tspec>
            <td:TspecIndex>1</td: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>
         </td:Tspec>
      </td: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>
         <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>
</TrafficDescriptorContainer>
```

2.2.32 A_ARG_TYPE_Layer2MappingContainer

A_ARG_TYPE_Layer2MappingContainer is a required state variable. This is a structure which contains zero or one *Layer2Mapping* structures. This structure allows for an optional *Layer2Mapping* to be passed as an input or output argument of an action. It is an output parameter for *AdmitTrafficQos()* and *UpdateAdmittedQos*() used to return an optional *Layer2Mapping*.

2.2.32.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument MUST validate against the XML schema for `Layer2MappingContainer` in the XML namespace `"urn:schemas-upnp-org:qos:Layer2MappingContainer"` which is located at `"http://www.upnp.org/schemas/qos/Layer2MappingContainer-v3.xsd"`.

2.2.32.2 Description of fields in the A_ARG_TYPE_Layer2Mapping Container state variable

This structure allows for an optional *Layer2Mapping* to be passed as an input or output argument of an action.

Layer2Mapping: This optional field contains a *Layer2Mapping*. Refer to clause 2.2.24 for more details on *Layer2Mapping*.

2.2.32.3 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<Layer2MappingContainer xsi:schemaLocation="urn:schemas-upnp-
org:qos:Layer2MappingContainer Layer2MappingContainer-v3.xsd" xmlns="urn:schemas-
upnp-org:qos:Layer2MappingContainer" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:lm="urn:schemas-upnp-org:qos:Layer2Mapping">
   <Layer2Mapping>
      <lm:TrafficHandle>TH0ac4f92a23a</lm:TrafficHandle>
      <lm:QosSegmentId>174A0AC77D0A872C3A</lm:QosSegmentId>
      <lm:Layer2StreamId>
00000000000000000000000000000000000000000000000000000382d2a1
      </lm:Layer2StreamId>
   </Layer2Mapping>
</Layer2MappingContainer>
```

2.2.33 A_ARG_TYPE_QosDeviceInfoContainer

A_ARG_TYPE_QosDeviceInfoContainer is a required state variable. This is a structure which contains zero or one *QosDeviceInfo* structures. This structure allows for an optional *QosDeviceInfo* structure to be passed as an input or output argument of an action.

2.2.33.1 XML Schema Definition

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

2.2.33.2 Description of fields in the A_ARG_TYPE_QosDeviceInfo Container state variable

This structure allows for an optional *QosDeviceInfo* to be passed as an input or output argument of an action.

QosDeviceInfo: This optional field contains a *QosDeviceInfo*. Refer to clause 2.2.9 for more details on *QosDeviceInfo*.

2.2.33.3 Sample argument XML string

```
<?xml version="1.0" encoding="UTF-8"?>
<QosDeviceInfoContainer xsi:schemaLocation="urn:schemas-upnp-
org:qos:QosDeviceInfoContainer QosDeviceInfoContainer-v3.xsd" xmlns="urn:schemas-
upnp-org:qos:QosDeviceInfoContainer"
xmlns:qdi="http://www.upnp.org/schemas/QosDeviceInfo.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <QosDeviceInfo>
      <qdi:TrafficHandle>TH1b4-a222-08002b34c0037f921234-723c-
11b4</qdi:TrafficHandle>
      <qdi:SourcePort>554</qdi:SourcePort>
      <qdi:DestinationPort>4712</qdi:DestinationPort>
      <qdi:IpProtocol>17</qdi:IpProtocol>
   </QosDeviceInfo>
</QosDeviceInfoContainer>
```

2.3 Eventing and Moderation

Table 2-6 — Event Moderation

Variable Name	Evented	Moderated Event	Max Event Rate [a]	Logical Combi-nation	Min Delta per Event [b]
A_ARG_TYPE_AdmitTrafficQosExtendedResult	No	No	N/A	N/A	N/A
A_ARG_TYPE_AdmitTrafficQosSucceeded	No	No	N/A	N/A	N/A
A_ARG_TYPE_ConfRotameterObservations	No	No	N/A	N/A	N/A
A_ARG_TYPE_Layer2Mapping	No	No	N/A	N/A	N/A
A_ARG_TYPE_Layer2MappingContainer	No	No	N/A	N/A	N/A
A_ARG_TYPE_ListOfAdmittedTraffic	No	No	N/A	N/A	N/A
A_ARG_TYPE_ListOfMostRecentUnexpectedStreamChanges	No	No	N/A	N/A	N/A
A_ARG_TYPE_MaxPossibleRotameterObservations	No	No	N/A	N/A	N/A
A_ARG_TYPE_NewTrafficLeaseTime	No	No	N/A	N/A	N/A
A_ARG_TYPE_NumberOfUnexpectedStreamChangesReported	No	No	N/A	N/A	N/A
A_ARG_TYPE_NumberOfUnexpectedStreamChangesRequested	No	No	N/A	N/A	N/A
A_ARG_TYPE_NumRotameterObservations	No	No	N/A	N/A	N/A
A_ARG_TYPE_NumTrafficDescriptors	No	No	N/A	N/A	N/A
A_ARG_TYPE_PreemptingTrafficInfo	No	No	N/A	N/A	N/A
A_ARG_TYPE_PreferredQph	No	No	N/A	N/A	N/A
A_ARG_TYPE_QosDeviceCapabilities	No	No	N/A	N/A	N/A
A_ARG_TYPE_QosDeviceExtendedState	No	No	N/A	N/A	N/A
A_ARG_TYPE_QosDeviceInfo	No	No	N/A	N/A	N/A
A_ARG_TYPE_QosDeviceInfoContainer	No	No	N/A	N/A	N/A
A_ARG_TYPE_QosDeviceState	No	No	N/A	N/A	N/A
A_ARG_TYPE_QosStateId	No	No	N/A	N/A	N/A
A_ARG_TYPE_Resource	No	No	N/A	N/A	N/A
A_ARG_TYPE_RotameterInformation	No	No	N/A	N/A	N/A
A_ARG_TYPE_SetPreferredQphResults	No	No	N/A	N/A	N/A
A_ARG_TYPE_TrafficDescriptor	No	No	N/A	N/A	N/A
A_ARG_TYPE_TrafficDescriptorContainer	No	No	N/A	N/A	N/A
A_ARG_TYPE_TrafficDescriptorsPerInterface	No	No	N/A	N/A	N/A
A_ARG_TYPE_TrafficDescriptorsWanted	No	No	N/A	N/A	N/A
A_ARG_TYPE_TrafficHandle	No	No	N/A	N/A	N/A
MostRecentStreamAction	Yes	Yes	2	N/A	N/A
PathInformation	Yes	Yes	30	N/A	N/A
UnexpectedStreamChange	Yes	Yes	2	N/A	N/A

[a] Determined by N, where Rate = one Event every N secs.

[b] (N) * (allowedValueRange Step).

2.3.1 Event Model

PathInformation: The state variable *PathInformation* is required and MUST be evented.

When there is a change in *PathInformation*, the *QosDevice* Service will issue an event and send the updated *PathInformation* variable in the body of the event. This event is moderated to avoid flooding the network with repeated events.

MostRecentStreamAction: The *MostRecentStreamAction* state variable is optional, but MUST be evented when implemented. If the *GetUnexpectedStreamChanges()* action is implemented this event MUST be implemented.

Any time a *SetupTrafficQos()*, *ReleaseTrafficQos()*, *AdmitTrafficQos()*, *UpdateAdmittedQos()* **or** *ReleaseAdmittedQos()* action is invoked successfully the *QosDevice* Service will issue an event and send the updated *MostRecentStreamAction* variable in the body of the event. This event is moderated to avoid flooding the network with repeated events.

UnexpectedStreamChange: The state variable *UnexpectedStreamChange* is required and MUST be evented. The *QosDevice* Service MUST increment the *UnexpectedStreamChange* field and send it in the body of the event whenever one of the following events occurs:

The *PreemptingTrafficInfo* structure in a *ReleaseAdmittedQos()* action indicates that preemption has occurred.

The *PreemptingTrafficInfo* structure in a *UpdateAdmittedQos()* action indicates that preemption has occurred.

The *QosDevice* Service detects that the L2 Technology has spontaneously—i.e., without initiation by the *QosDevice* Service—either released or changed the stream.

This event is moderated to avoid flooding the network with repeated events.

2.4 Actions

Immediately following Table 2-7 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. [a]
GetQosDeviceCapabilities()	R
GetQosState()	R
SetupTrafficQos()	R
ReleaseTrafficQos()	R
GetPathInformation()	R
GetQosDeviceInfo()	O
GetRotameterInformation()	O
ConfigureRotameterObservation()	O
AdmitTrafficQos()	R
UpdateAdmittedQos()	R
ReleaseAdmittedQos()	R
GetExtendedQosState()	R
SetPreferredQph()	O
GetUnexpectedStreamChanges()	O
VerifyTrafficHandle()	O
UpdateTrafficLeaseTime()	R
SetL2Map()	O
[a] R = Required, O = Optional, X = Non-standard	

The following Actions are required (except *GetQosDeviceInfo()*) to support backward compatibility with devices developed to earlier versions of UPnP-QoS, but their use by Control Points designed for UPnP-QoS v3 is discouraged.

- *GetQosDeviceCapabilities()* (the recommended action is *GetExtendedQosState()*)
- *GetQosState()* (the recommended action is *GetExtendedQosState()*)
- *SetupTrafficQos()* (the recommended action is *AdmitTrafficQos()*)
- *ReleaseTrafficQos()* (the recommended action is *ReleaseAdmittedQos()*)
- *GetQosDeviceInfo()* (the recommended action is *GetExtendedQosState()*)

2.4.1 GetQosDeviceCapabilities()

This action is required to support backward compatibility with devices developed to earlier versions of UPnP-QoS, but its use by Control Points designed for UPnP-QoS v3 is discouraged. Use of *GetExtendedQosState()* is recommended.

This action returns the static QoS capabilities of the *QosDevice* Service.

2.4.1.1 Arguments

Table 2-8 — Arguments for *GetQosDeviceCapabilities()*

Argument	Direction	relatedStateVariable
QosDeviceCapabilities	Out	*A_ARG_TYPE_QosDeviceCapabilities*

2.4.1.2 Service requirements

None.

2.4.1.3 Control Point requirements when calling the action

None.

2.4.1.4 Dependency on State (if any)

None, these are static capabilities.

2.4.1.5 Effect on State (if any)

None.

2.4.1.6 Errors

Refer to UPnP Device architecture for common error codes.

Table 2-9 — Error Codes for *GetQosDeviceCapabilities()*

errorCode	errorDescription	Description

2.4.2 GetQosState()

This action is required to support backward compatibility with devices developed to earlier versions of UPnP-QoS, but its use by Control Points designed for QoS v3 is discouraged. Use of *GetExtendedQosState()* is recommended.

The *GetQosState()* action returns the instantaneous QoS state of the device. It does not provide complete information about the state of the v3 *QosDevice* Service. The device MUST list only the *TrafficDescriptor*(s) that were registered in the device by use of the *SetupTrafficQos()* action in the *ListOfTrafficDescriptors* argument.

2.4.2.1 Arguments

Table 2-10 — Arguments for *GetQosState()*

Argument	Direction	relatedStateVariable
QosDeviceState	Out	*A_ARG_TYPE_QosDeviceState*
NumberOfTrafficDescriptors	Out	*A_ARG_TYPE_NumTrafficDescriptors*
ListOfTrafficDescriptors	Out	*A_ARG_TYPE_TrafficDescriptorsPerInterface*

2.4.2.2 Service requirements

The device MUST list only the *TrafficDescriptor*(s) that were registered in the device by use of the *SetupTrafficQos()* or *AdmitTrafficQos()* actions in the *ListOfTrafficDescriptors* argument.

2.4.2.3 Control Point requirements when calling the action

None.

2.4.2.4 Dependency on State (if any)

This action does not have any dependency on the state of *QosDevice* Service.

2.4.2.5 Effect on State (if any)

This action does not have any effect on the state of *QosDevice* Service.

2.4.2.6 Errors

Table 2-11 — Error Codes for *GetQosState()*

errorCode	errorDescription	Description

2.4.3 SetupTrafficQos()

This action is required to support backward compatibility with devices developed to earlier versions of UPnP-QoS. While it can be used to set up a Prioritized Traffic Stream, its use by Control Points designed for UPnP-QoS v3 is discouraged. Use of *AdmitTrafficQos()* is recommended.

The *SetupTrafficQos()* action indicates to the device to set up Prioritized QoS for the Traffic described by *SetupTrafficDescriptor*.

Please refer to Annex A 'Traffic Descriptor Matrix' in the [QM] document for information about all of the fields of the *TrafficDescriptor* and how they are used.

2.4.3.1 Arguments

Table 2-12 — Arguments for *SetupTrafficQos()*

Argument	Direction	relatedStateVariable
SetupTrafficDescriptor	In	*A_ARG_TYPE_TrafficDescriptor*
QosStateId	In	*A_ARG_TYPE_QosStateId*

2.4.3.2 Service requirements

The *QosDevice* Service MUST ignore the *RequestedQosType* field in *SetupTrafficDescriptor*.

If the *QosStateId* input argument does not match the current QoS State ID of the *QosDevice* Service, it MUST return error 760.

If the *QosDevice* Service can determine that it is not on the path for this traffic stream and determines that this device is not on the path, it MUST return error 751.

If there is no Traffic Descriptor registered in the *QosDevice* Service with the same *TrafficHandle*, then this TrafficDescriptor will be registered in the *QosDevice* Service after the successful execution of this action.

If the device already has the Traffic Descriptor (identified by the *TrafficHandle*) registered, then the *QosDevice* Service MUST return an error 702.

If the *QosDevice* Service does not receive a Traffic Descriptor with a *TrafficImportanceNumber*, the *QosDevice* Service MUST return error 711.

If the *QosDevice* Service does not receive a Traffic Descriptor with *ActiveTspecIndex*, it MUST return error 711.

If the *QosDevice* Service does not receive a Traffic Descriptor with a *TrafficHandle*, or *TrafficHandle* has a NULL value, it MUST return error 700.

In the Traffic Descriptor to the *QosDevice* Service, the TSPEC for which Traffic Policy is provided is indicated by the *ActiveTspecIndex*. *ActiveTspecIndex* MUST be one of the *TspecIndex* values in the *AvailableOrderedTspecList*. If not, *QosDevice* Service MUST return the error 720.

If the *TrafficId* in the Traffic Descriptor is incomplete, the *QosDevice* Service MUST return error 710. The *TrafficId* MUST include a *SourceAddress*, *DestinationAddress*, *SourcePort*, *DestinationPort* and *IpProtocol* to be complete.

2.4.3.3 Control Point requirements when calling the action

A Control Point (i.e., *QosManager*) MUST supply the *TrafficImportanceNumber* in Traffic Descriptor to *QosDevice* Service when calling the *SetupTrafficQos()* action.

A Control Point (i.e., QosManager) MUST supply the *ActiveTspecIndex* in Traffic Descriptor to *QosDevice* Service when calling the *SetupTrafficQos()* action.

A Control Point (i.e., QosManager) MUST supply the *TrafficHandle* in Traffic Descriptor to *QosDevice* Service when calling the *SetupTrafficQos()* action.

A Control Point (i.e., QosManager) MUST supply an *ActiveTspecIndex* that is one of the *TspecIndex* values in the *AvailableOrderedTspecList* in Traffic Descriptor to *QosDevice* Service when calling the *SetupTrafficQos()* action.

A Control Point (i.e., *QosManager*) MUST supply a complete *TrafficId* that includes a *SourceAddress*, *DestinationAddress*, *SourcePort*, *DestinationPort* and *IpProtocol*.

2.4.3.4 Dependency on State (if any)

QosStateId is provided as an input to this action. In case the current *QosStateId* of the device is different than the one specified by the Control Point (i.e., *QosManager*), the action returns the error 760. Otherwise, the *QosDevice* Service sets up QoS for the traffic stream.

2.4.3.5 Effect on State (if any)

Upon successful completion of this action, the *QosDevice* Service sets up QoS for the traffic specified in the action request. Please refer to ‘Theory of Operation’ clause for more details.

The *QosDevice* Service MUST NOT modify any of the elements assigned by the *QosManager* in the *SetupTrafficDescriptor* structure. Upon successful completion of *SetupTrafficQos()*, source devices implementing the *QosDevice* Service MUST prioritize the traffic, associated with the *TrafficId*, according to the *TrafficImportanceNumber* (hence *PacketTaggingSupported*=“Yes”) on their output interfaces. Intermediate devices implementing the *QosDevice* Service with *PacketTaggingSupported* set to "Yes" MUST prioritize the traffic associated with the *TrafficId* according to the *TrafficImportanceNumber* on their output interfaces irrespective of incoming traffic priority.

2.4.3.6 Errors

Table 2-13 — Error Codes for *SetupTrafficQos()*

errorCode	errorDescription	Description
700	*TrafficHandle* missing or empty	*TrafficHandle* MUST be filled in as input to this action.
702	*TrafficHandle* already registered	A Control Point (i.e., QosManager) is not allowed to set up or modify QoS using *SetupTrafficQos()* if QoS has already been set up for that handle.
710	Incomplete *TrafficId*	All *TrafficId* fields (*SourceAddress*, *DestinationAddress*, *SourcePort*, *DestinationPort* and *IpProtocol*) MUST be present.
711	Insufficient information	The input information is not complete.
716	An input parameter (e.g. Traffic Descriptor) does not validate against the XML schema	One of the XML-based input arguments does not follow the schema
720	*ActiveTspecIndex* is not a *TspecIndex*	
751	Device not on path	
760	*QosStateId* does not match	Please refer to the 'Theory of Operation' clause.
761	*QosDevice* Service cannot set up this stream	QoS Setup failed, e.g device does not support prioritized QoS

2.4.4 ReleaseTrafficQos()

This action is required to support backward compatibility with devices developed to earlier versions of UPnP-QoS. While it can be used to release a prioritized Traffic Stream, its use by Control Points designed for UPnP-QoS v3 is discouraged. Use of *ReleaseAdmittedQos()* is recommended.

The *ReleaseTrafficQos()* action indicates that the traffic stream is no longer managed by UPnP-QoS at this device. The *ReleaseTrafficQos()* action provides an indication to the device to release the QoS for the traffic identified by *ReleaseTrafficHandle*.

This action will cause all QoS for this *TrafficHandle* on this *QosDevice* Service to be released, unlike *ReleaseAdmittedQos()* which accepts a Resource input argument to specify for which Resource the QoS reservation should be released.

2.4.4.1 Arguments

Table 2-14 — Arguments for *ReleaseTrafficQos()*

Argument	Direction	relatedStateVariable
ReleaseTrafficHandle	In	*A_ARG_TYPE_TrafficHandle*

2.4.4.2 Service requirements

The *QosDevice* Service MUST return an error code 703 if the input *ReleaseTrafficHandle* is not valid. An input *ReleaseTrafficHandle* is valid only if it is part of one and only one of the TrafficDescriptors stored in that device.

The *QosDevice* Service MUST return an error code 717 if *ReleaseTrafficQos()* is invoked for a parameterized stream.

2.4.4.3 Control Point requirements when calling the action

The Control Point MUST supply a valid traffic handle to revoke the QoS of the traffic stream.

2.4.4.4 Dependency on State (if any)

The *ReleaseTrafficHandle* provided has to be valid and known to the *QosDevice* Service.

2.4.4.5 Effect on State (if any)

After this call, *ReleaseTrafficHandle* is no longer registered at the device to provide QoS. The device MUST release all its QoS resources allocated to that traffic.

2.4.4.6 Errors

Table 2-15 — Error Codes for *ReleaseTrafficQos()*

errorCode	errorDescription	Description
703	*TrafficHandle* unknown to this device	The *TrafficHandle* is unknown to this device
717	Illegal action on this *TrafficHandle*	An action that is illegal for this *TrafficHandle* has been invoked. E.g., *ReleaseTrafficQos()* invoked for a parameterized stream.

2.4.5 GetPathInformation

This is a required action. This action returns the *PathInformation* structure for that *QosDevice* Service providing information about the reachable MAC addresses. This information is used by the Control Point (i.e., *QosManager*) for path determination.

2.4.5.1 Arguments

Table 2-16 — Arguments for *GetPathInformation()*

Argument	Direction	relatedStateVariable
PathInformation	Out	*PathInformation*

2.4.5.2 Service requirements

None.

2.4.5.3 Control Point requirements when calling the action

None.

2.4.5.4 Dependency on State (if any)

None.

2.4.5.5 Effect on State (if any)

None.

2.4.5.6 Errors

Table 2-17 — Error Codes for Get*PathInformation*

errorCode	errorDescription	Description

2.4.6 GetQosDeviceInfo()

This is an optional action to support Control Points (e.g. UPnP-AV) in setting up QoS. A general UPnP-AV control point only knows the SourceUuid (the UUID of the MediaServer) and the DestinationUuid (the UUID of the MediaRenderer). When supported, this action returns the *QosDeviceInfo* structure providing transport-related information specific to the stream identified by the *TrafficDescriptor* information provided. The information returned may include port numbers used, source and destination IP addresses and IP protocol. Use of *GetQosDeviceInfo()* by Control Points designed for QoS v3 is discouraged. Use of *GetExtendedQosState()* is recommended.

The *QosDevice* Service at the source or sink can determine this information based on the following:

- Available elements of the *TrafficId* (e.g., SourceUuid, DestinationUuid or source and destination IP addresses)
- *AvTransportUri* and *AvTransportInstanceId* if specified
- *MediaServerConnectionId* and *MediaRendererConnectionId* if specified

QosDeviceInfo returned as part of this action is used by the *QosManager* to complete the traffic identifier structure.

2.4.6.1 Arguments

Table 2-18 — Arguments for *GetQosDeviceInfo()*

Argument	Direction	relatedStateVariable
TrafficDescriptor	In	*A_ARG_TYPE_TrafficDescriptor*
QosDeviceInfo	Out	*A_ARG_TYPE_QosDeviceInfo*

2.4.6.2 Service requirements

If the *QosDevice* Service receives information which is insufficient to determine the port numbers and protocol the *QosDevice* Service will return error 712.

2.4.6.3 Control Point requirements when calling the action

The Control Point (i.e., *QosManager*) should supply all available information related to the UPnP AV scenario such as *MediaServerConnectionId*, *MediaRendererConnectionId*, *AvTransportUri* or *AvTransportInstanceId*.

2.4.6.4 Dependency on State (if any)

None

2.4.6.5 Effect on State (if any)

None

2.4.6.6 Errors

Table 2-19 — Error Codes for *GetQosDeviceInfo()*

errorCode	errorDescription	Description
712	Incomplete information to determine protocol and port numbers	Incomplete information. For example, in case of the UPnP AV scenario, MediaServerConnectionId, MediaRendererConnectionId, AvTransportUri or AvTransportInstanceId is required but not provided.

2.4.7 ConfigureRotameterObservation()

2.4.7.1 Arguments

Table 2-20 — Arguments for *ConfigureRotameterObservation()*

Argument	Direction	relatedStateVariable
RequestedConfRotameterObservations	In	*A_ARG_TYPE_ConfRotameterObservations*
MaxPossibleRotameterObservations	Out	*A_ARG_TYPE_MaxPossibleRotameterObservations*

2.4.7.2 Service requirements

RequestedConfRotameterObservations is provided as an input to this action. In case the *RequestedConfRotameterObservations* is more than the capabilities of the device, the action returns an error (Error Code 730 or 731, described below). If *ROPeriod* is greater than *MonitorResolutionPeriod*, the action returns error code 734. If no error is returned, observations MUST begin immediately, i.e., the device MUST not wait until a Control Point requests a Rotameter observation.

If a Control Point specifies a list of *Layer2StreamIds* in the *RequestedConfRotameterObservations* argument, and the *QosDevice* Service isn't capable of providing diagnostic information on per stream basis, the action returns an error with Error Code 719.

If a Control Point specifies *ROAddr* in the *RequestedConfRotameterObservations* argument, and the *QosDevice* Service doesn't support *ROAddr* configuration, the action MUST return an error with Error Code 736.

If a Control Point specifies more *Layer2StreamIds* than the device is capable of reporting, the *QosDevice* Service MUST provide information on as many *Layer2StreamIds* as it can.

Because the Rotameter service is purposed at providing diagnostic value (e.g. which device or stream is sending or receiving how much traffic on the network), and most applications do not have steady traffic characteristics, it is recommended that *ROPeriod* and *MonitorResolutionPeriod* are the same value with sufficiently small granularity (1 second for example).

Upon successful configuration of the *QosDevice* Service Rotameter function, the maximum number of observations the device is capable of reporting MUST be returned, i.e., *MaxPossibleRotameterObservations*. This allows a Control Point to know the maximum number of observations to request when calling *GetRotameterInformation()*.

2.4.7.3 Dependency on State (if any)

None

2.4.7.4 Control Point requirements when calling the action

A Control Point (*QosManager*) calling action *GetRotameterInformation()* should ensure that a *QosDevice* Service has been configured with *ConfigureRotameterObservation()*.

2.4.7.5 Effect on State (if any)

Upon successful invocation of this action, the *QosDevice* Service will begin monitoring network traffic as requested.

2.4.7.6 Errors

Table 2-21 — Error Codes for *ConfigureRotameterObservation()*

errorCode	errorDescription	Description
730	*ROPeriod* incapable	*ROPeriod* is outside the capabilities of the device.
731	*MonitorResolutionPeriod* incapable	*MonitorResolutionPeriod* is outside the capabilities of the device.
734	Invalid Arguments	*ROPeriod* > *MonitorResolutionPeriod*
719	Stream specific information not supported	Information on per stream basis cannot be provided.
736	*ROAddr* configuration not supported	*ROAddr* configuration not supported.

2.4.8 GetRotameterInformation()

This is an optional action. When supported, this action call returns the *RotameterInformation* structure for that *QosDevice* Service providing information about the reachable MACs and Rotameter information. This information may be used directly by any Control Point to observe traffic flow. The *QosDevice* Service provides the most recent (in time) observations indicated by the number *RequestedNumRotameterObservations*.

2.4.8.1 Arguments

Table 2-22 — Arguments for *GetRotameterInformation()*

Argument	Direction	relatedStateVariable
RequestedNumRotameterObservations	In	*A_ARG_TYPE_NumRotameterObservations*
RotameterObservation	Out	*A_ARG_TYPE_RotameterInformation*

2.4.8.2 Service requirements

A *QosDevice* Service MUST be first configured using the action *ConfigureRotameterObservation()* before accessing the *RotameterObservation* using the *GetRotameterInformation()* action. If this sequence is not followed then the *QosDevice* Service MUST return the error 735.

2.4.8.3 Control Point requirements when calling the action

A Control Point MUST first configure the *QosDevice* Service by calling the action *ConfigureRotameterObservation()* before accessing the *RotameterObservation*.

2.4.8.4 Dependency on State (if any)

RequestedNumRotameterObservations is provided as an input to this action to indicate the number of Rotameter Observations per MAC address requested by a Control Point. The response to *GetRotameterInformation()* provides the most recent (in time) observations indicated by the number *RequestedNumRotameterObservations*. In case the *RequestedNumRotameterObservations* is more than the Rotameter service is capable of providing, the action MUST return error code 732. In case the *RequestedNumRotameterObservations* is more than the Rotameter service currently has at this time, the action MUST return error code 733. It is assumed that the Control Point's request will be consistent with the information given during the invocation of *ConfigureRotatmeterObservations()*. Otherwise, the *QosDevice* Service returns *RotameterObservation*.

2.4.8.5 Effect on State (if any)

None

2.4.8.6 Errors

Table 2-23 — Error Codes for *GetRotameterInformation()*

errorCode	errorDescription	Description
732	Requested too many observations	RequestedNumRotameterObservations is more than device capabilities
735	*ConfigureRotameterObservation()* has not been invoked	*ConfigureRotameterObservation()* has not been invoked before calling GetRotameterObservation
733	No valid observation	Unable to provide an observation at this time.

2.4.9 AdmitTrafficQos()

The *AdmitTrafficQos()* action causes the device to invoke the underlying Admission Mechanism to set up for the traffic described by *AdmitTrafficDescriptor* on the resource specified in *Resource*. This action can be used to set up either a prioritized or parameterized traffic stream. Its use by Control Points designed for UPnP-QoS v3 is recommended.

2.4.9.1 Arguments

Table 2-24 — Arguments for *AdmitTrafficQos()*

Argument	Direction	relatedStateVariable
AdmitTrafficDescriptor	In	*A_ARG_TYPE_TrafficDescriptor*
Resource	In	*A_ARG_TYPE_Resource*
AdmitTrafficQosSucceeded	Out	*A_ARG_TYPE_AdmitTrafficQosSucceeded*
AdmitTrafficQosExtendedResult	Out	*A_ARG_TYPE_AdmitTrafficQosExtendedResult*
Layer2MappingContainer	Out	*A_ARG_TYPE_Layer2MappingContainer*

2.4.9.2 Service requirements

If the *QosDevice* Service does not receive an *AdmitTrafficDescriptor* with an *ActiveTspecIndex*, it MUST return error 711.

If the *QosDevice* Service does not receive a *AdmitTrafficDescriptor* with a *TrafficHandle*, or *TrafficHandle* has a NULL value, it MUST return error 700.

In the *AdmitTrafficDescriptor* to the *QosDevice* Service, the *TSPEC* for which admission is requested is indicated by the *ActiveTspecIndex*. *ActiveTspecIndex* MUST be one of the *TspecIndex* values in the *AvailableOrderedTspecList*. If an *ActiveTspecIndex* is not provided, the *QosDevice* Service MUST return the error 720.

If there is no Traffic Descriptor registered in the *QosDevice* Service with the same *TrafficHandle* for the indicated *Resource*, then this Traffic Descriptor MUST be registered in the *QosDevice* Service after the successful execution of this action results in resources being allocated (return argument *AdmitTrafficQosSucceeded* = "**1**").

If the device already has the *AdmitTrafficDescriptor* (identified by the *TrafficHandle*) registered with the same Resource, then the *QosDevice* Service MUST return an error 717.

If the *RequestedQosType* field is set to "0" (Prioritized QoS) or is not specified, the *QosDevice* Service MUST set up the stream as prioritized.

The *QosDevice* Service MUST return Successful completion with *Reason* "001" and *AdmitTrafficQosSucceeded* of "**1**" and MUST NOT reserve resources on the underlying L2 Technology if both of these conditions are met:

- The *RequestedQosType* field is set to "2" (Parameterized QoS) or "1" (Hybrid QoS) and the *Resource* supports parameterized QoS
- And the *QosDevice* Service determines that it is not responsible for reserving resource on that QoS Segment. This is based on the L2 technology for the Interface and the values of the fields *QDDownstream* and *QDUpstream* in the *Resource* argument. See [QD_Add]

If the *RequestedQosType* field is set to "1" (Hybrid QoS) and the QoS Segment does not support parameterized QoS, the *QosDevice* Service MUST set up the stream as prioritized.

If the *RequestedQosType* field is set to "1" (Hybrid QoS) and the QoS Segment does support parameterized QoS, the *QosDevice* Service MUST set up the stream as parameterized.

If the *QosDevice* Service is able to learn the *Layer2StreamId* of the admitted stream, it SHOULD populate the *Layer2MappingContainer* with this information.

If the *RequestedQosType* field is set to "2" (Parameterized QoS), the *QosDevice* Service MUST set up the stream as Parameterized. If the request is on a QoS Segment that does not support parameterized QoS then the *QosDevice* Service MUST return error code 764.

If the *RequestedQosType* field is set to "1" (Hybrid QoS) or "2" (Parameterized QoS) and the stream is successfully admitted, the resources reserved for the stream MUST be sufficient to support the active TSPEC.

If the *RequestedQosType* field is set to a value greater than "2", the *QosDevice* Service MUST return error 718.

If the *Resource* argument contains a *DeviceResource* or *NetworkResource* that is not known to the *QosDevice* Service, it MUST return error 713.

If *AdmitTrafficQos()* is unable to admit the requested *AdmitTrafficDescriptor*, it SHOULD populate the *ListOfLayer2StreamIds* in the *AdmitTrafficQosExtendedResults* output argument with all blocking *Layer2StreamIds* on the same QoS Segment. It may not be possible for the *QosDevice* Service to determine all *Layer2StreamIds*.

If *AdmitTrafficQos()* is unable to admit the request due to lack of resources, it MUST return Success with the *Reason* set to "762".

If the *QosDevice* Service can determine that the indicated *NetworkResource* is not on the stream's path, it MUST return error 751.

If the *TrafficId* in *AdmitTrafficDescriptor* is incomplete, the *QosDevice* Service MUST return error 710. A complete *TrafficId* includes *SourceAddress*, *SourcePort*, *DestinationAddress*, *DestinationPort* and *IpProtocol*.

If *AdmitTrafficDescriptor* contains a *TrafficHandle* that is already registered on this *QosDevice* Service for a different Resource, then *AdmitTrafficDescriptor* and the previously registered Traffic Descriptor MUST be equal with the exception of the segment-specific information (i.e., *QosSegmentSpecificParameters* within the TSPEC field of the Traffic Descriptors). Otherwise the *QosDevice* Service MUST return error 767.

If the *RequestedQosType* field is set to "1" (Hybrid QoS) or "2" (Parameterized) and the *AdmitTrafficDescriptor* does not contain a *TrafficLeaseTime*, then the *QosDevice* Service MUST return error 791.

If the underlying L2 Technology is able to report a value for *MaxCommittedDelay* then the *QosDevice* Service that is responsible for reserving resources with the underlying L2 Technology for the requested interface MUST populate the *MaxCommittedDelay* field in the *AdmitTrafficQosExtendedResult* return argument upon successful admission when *RequestedQosType* field is set to “1” (Hybrid QoS) or “2” (Parameterized QoS). See clause 2.4.9.2.1 for information on calculating *MaxCommittedDelay.*

If the underlying L2 Technology is able to report a value for *MaxCommittedJitter* then the *QosDevice* Service that is responsible for reserving resources with the underlying L2 Technology for the requested interface MUST populate the *MaxCommittedJitter* field in the *AdmitTrafficQosExtendedResult* return argument upon successful admission when *RequestedQosType* field is set to “1” (Hybrid QoS) or “2” (Parameterized QoS).

If *QosSegmentMaxDelayHigh* is provided by the Control Point (i.e. a QosManager) and *QosSegmentMaxDelayLow* is not provided then QosDevices MUST provide any delay less than or equal to *QosSegmentMaxDelayHigh* for the segment. If both *QosSegmentMaxDelayHigh* and *QosSegmentMaxDelayLow* are provided then the *QosDevice* Service SHOULD provide the LOWEST possible value greater than or equal to *QosSegmentMaxDelayLow* and less than or equal to *QosSegmentMaxDelayHigh.*

If the TrafficDescriptor is successfully admitted, then the QosDevice Service MUST return success with *Reason* “000”.

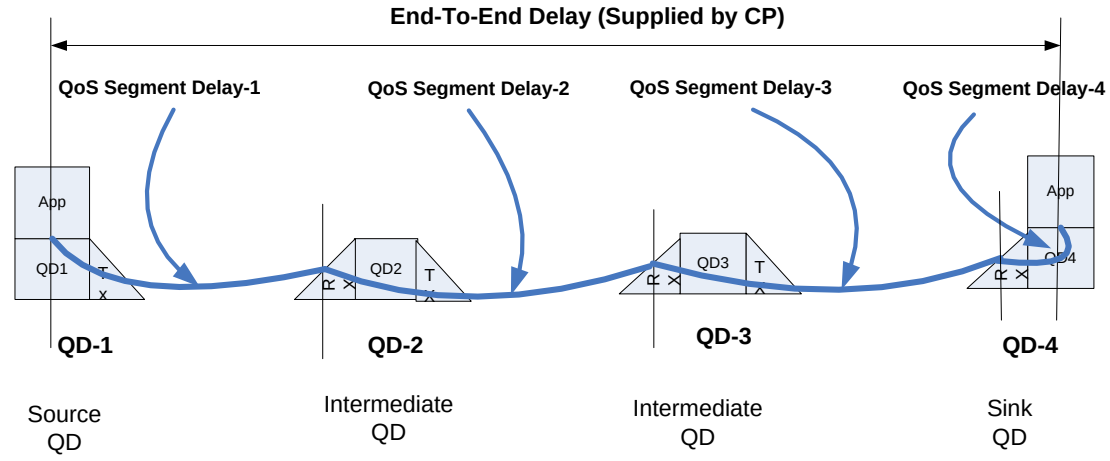
2.4.9.2.1 How to Calculate *MaxCommittedDelay*

The values returned in *MaxCommittedDelay* and *MaxCommittedJitter* represent the delay and jitter of either an individual *QosDevice* Service or of an entire QoS Segment. In the latter case, a technology that processes reservations serially through the QoS Segment will have the entire delay and jitter collected at either the source or sink end of the QoS Segment. These total values are reported by that *QosDevice* Service and all other *QosDevice* Services on the QoS Segment report zero or any delay or jitter component not reported by the source or sink device. In the first case, all *QosDevice* Services will report delay and jitter for only their portion of the QoS Segment. For each technology, a decision MUST be made as to how these values are going to be reported. All *QosDevice* Services for a technology MUST operate in the same manner to ensure proper reporting of these values as outlined in [QD_Add] .

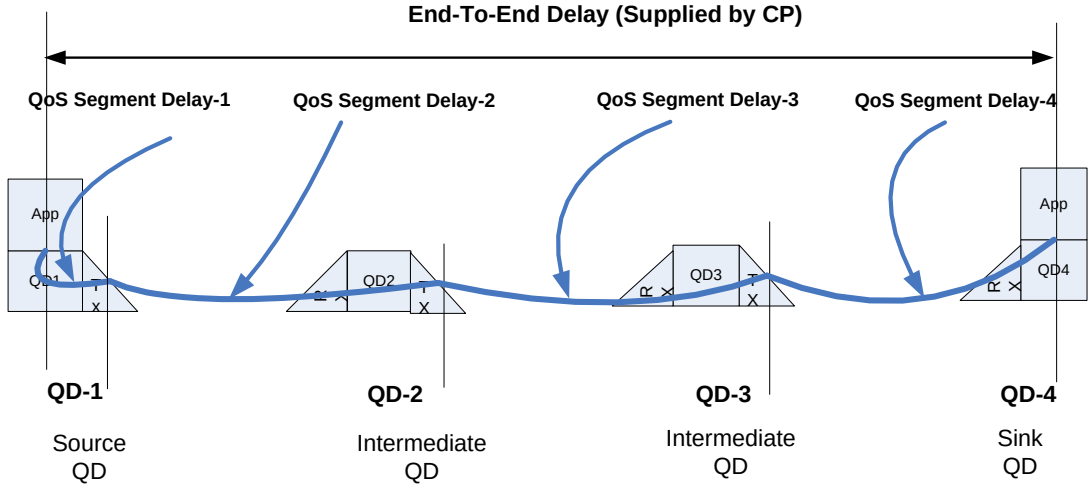
The QoS Segment Delay is defined as follows:

- For an intermediate device, the QoS Segment Delay is the delay introduced by that device for transmission or reception of a packet between itself and immediately adjacent devices (either upstream or downstream) in that QoS Segment. The precise definition is technology specific and can be found in the technology’s addendum.
- At the source *QosDevice* Service, the QoS Segment Delay is the time from the application submitting a block of data for sending until the source either starts sending the data packet at Layer 2 or the reception of the data packet by the immediately downstream device at Layer 2 in that segment. The precise definition is technology specific and can be found in the technology’s addendum.
- At the sink, the QoS Segment Delay is the time between the availability of data to the application and the beginning of transmission of a data packet by the most immediate upstream *QosDevice* Service or reception of the data packet by the sink device, at Layer 2. The precise definition is technology specific and can be found in the technology’s addendum.

The QoS Segment Delay is not necessarily constant and it will have a maximum and a minimum, for a period of observation.



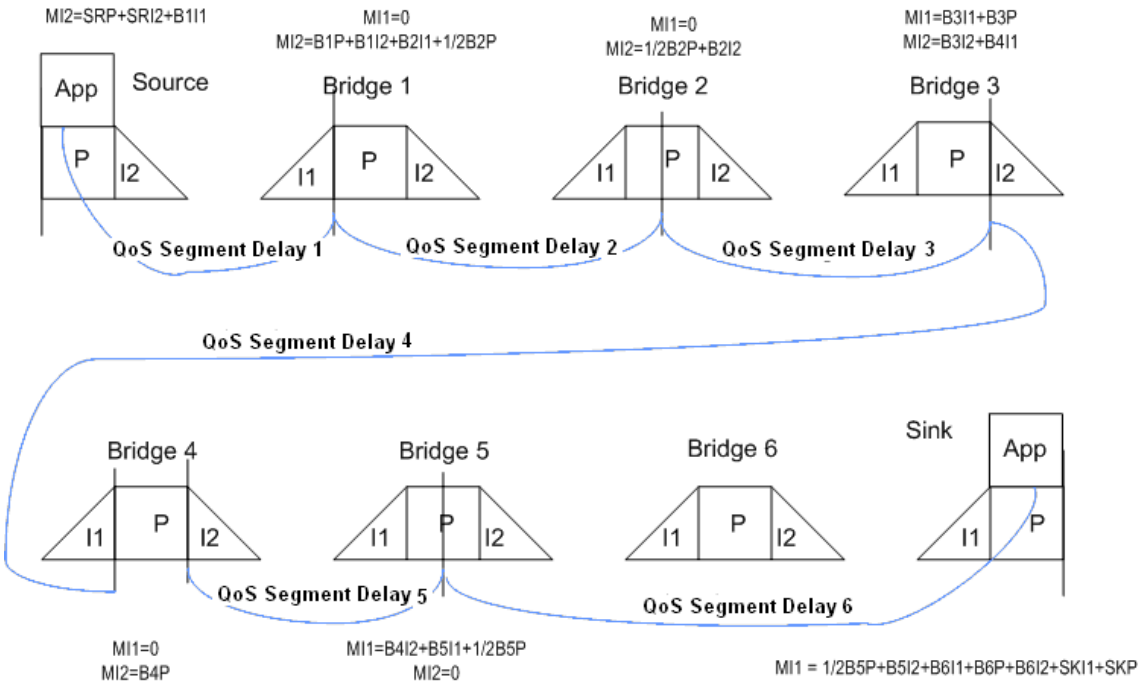
Example 1: A method for determining QoS Segment Delay values when delay is measured from transmitting device's perspective`



Example 2: A method for determining QoS Segment Delay values when delay is measured from receiving device's perspective

Figure 2-4 — Relationship between End-to-End Delay and QoS Segment Delay

The interfaces are graphically denoted by the triangles.



P = Qos Device Processing
I = Interface
B = Bridge
SR = Source
SK = Sink
M = MaxCommittedDelay

Source - Reports MaxCommittedDelay on I2 that represents delay for all of QoS Segment 1.
Bridge 1 - Reports MaxCommittedDelay on I2 that represents delay for all of QoS Segment 2 and reports MaxCommittedDelay of 0 on I1.
Bridge 2 - Reports MaxCommittedDelay on I2 that represents delay for Bridge 2 part of QoS Segment 3 and reports MaxCommittedDelay of 0 on I1.
Bridge 3 - Reports MaxCommittedDelay on I1 that represents delay for Bridge 3 part of QoS Segment 3, and MaxCommittedDelay on I2 that represents delay for all of Segment 4.
Bridge 4 - Reports MaxCommittedDelay on I2 which represents Processing delay on Bridge 4 and reports MaxCommittedDelay of 0 on I1.
Bridge 5 - Reports MaxCommittedDelay on I1 that represents delay for all of the QoS Segment 5 and reports MaxCommittedDelay of 0 on I2.
Bridge 6 - Is not a UPnP device. All delay must be reported via lower layers to the Sink.
Sink - Reports MaxCommittedDelay on I1 that represents delay for all of the QoS Segment 6

Figure 2-5 — Relationship between QoS Segment Delay And MaxCommittedDelay.

Calculation of *MaxCommittedDelay* for *QosDevice* Services and QoS Segments:

Within a source or sink *QosDevice* Service there are two components of delay used to calculate *MaxCommitedDelay*. First, there is delay associated with an interface of the *QosDevice* Service. Interface delay includes components such as communications and scheduling delay due to a network coordinator device. Second, there is delay associated with *QosDevice* Service processing outside the interface. A bridging *QosDevice* Service that connects QoS Segments has three components of delay: each of the two interfaces as well as processing delay. When reporting *MaxCommittedDelay* for an interface, the processing delay MUST be apportioned by the *QosDevice* Service between the interfaces in a manner that is invariant for that *QosDevice* Service. The L2 Technologies' delay returned to the *QosDevice* Service, is reported in one of three ways:

a) All of the delay associated with a device and device interface is reported directly by the *QosDevice* Service for that device via the return parameter *MaxCommittedDelay*. For bridge devices, all or part of the processing delay is reported on either interface as long as all delay is reported and no part of the delay is reported twice. Every *QosDevice* Service is responsible for reporting all of the delay associated with that device. No

component of delay of other devices on a common QoS Segment is reported by this *QosDevice* Service.

b) The delay associated with a QoS Segment is collected by one device on the QoS Segment using some lower layer communication method and reported by that device via its *QosDevice* Service. A device that reports delay for a QoS Segment and is the designated reporting device for that QoS Segment (source or sink of the QoS Segment as determined by the addendum for that technology) MUST report that delay via the *QosDevice* Service. Other devices that have reported delay components to the reporting device via some lower layer communications MUST NOT then report that delay as part of the *MaxCommittedDelay* value for the interface on the non reporting device.

c) A combination of the first two methods is used by a bridge device to report *MaxCommittedDelay* for an interface. On one interface of the bridge *MaxCommittedDelay* reports delay for the entire segment, as in b) above. On the other interface *MaxCommittedDelay* reports only the delay for the interface in the device, as in a) above. The processing delay for the Device is reported in one of two ways.

- All processing delay is reported as part of the *MaxCommittedDelay* value for one interface with zero reported for the other interface.
- Part of the processing delay is reported in the *MaxCommittedDelay* of one interface and the remainder is reported in the *MaxCommittedDelay* of the other interface

A device MUST ensure that all components of delay are reported by the *QosDevice* Service on some interface and that there is no duplication.

In all three cases, a device MUST ensure that all components of delay are reported by the QosDevice Service on some device and that there is no duplication.

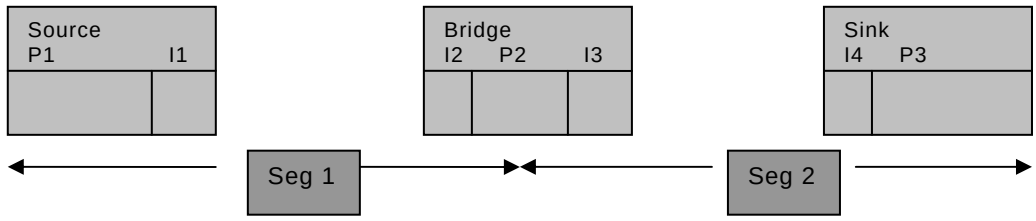


Figure 2-6 — Components of *MaxCommittedDelay*

Examples using Figure 2-6:

d) **Method 1:** All devices report only their own delay contributions for *MaxCommittedDelay*.

1) MaxCommittedDelay report by Source on I1 = P1 delay + I1 delay
2) MaxCommittedDelay report by Bridge on I2 = I2 delay + ½ P2 delay
3) MaxCommittedDelay report by Bridge on I3 = I3 delay + ½ P2 delay
4) MaxCommittedDelay report by Sink on I4 = I4 delay + P3 delay

e) **Method 2:** Segment 1 *MaxCommittedDelay* is reported by individual devices and Segment 2 *MaxCommittedDelay* is reported by the sink device

1) MaxCommittedDelay report by Source on I1 = P1 delay + I1 Delay
2) MaxCommittedDelay report by Bridge on I2 = I2 delay + ½ P2 delay
3) MaxCommittedDelay report by Bridge on I3 = zero

4) MaxCommittedDelay report by Sink on I4 = I2 delay + ½ P2 delay + I4 delay + P3 delay

f) **Method 3:** Segment 1 *MaxCommittedDelay* is reported by the source device and Segment 2 *MaxCommittedDelay* is reported by the sink device but the bridge device does not report its processing delay to the lower layers so it MUST report that as *MaxCommittedDelay* directly.

1) MaxCommittedDelay report by Source on I1 = P1 delay + I1 delay + I2 delay
2) MaxCommittedDelay report by Bridge on I2 = ½ P2 delay
3) MaxCommittedDelay report by Bridge on I3 = ½ P2 delay
4) MaxCommittedDelay report by Sink on I4 = I2 delay + I4 delay + P3 delay

In all examples, the apportionment ½ of P2 delay to either side of the bridge is arbitrary. The *QosDevice* Service has the option of reporting all of the processing delay on one interface and zero on the other or some fraction of each that adds up to the entire processing delay.

For example: If the entire delay across the bridge was 2ms then each interface could report 1ms or one interface could report 2ms and the other 0ms.

The *QosDevice* Sevice is required to follow the similar set of rules for reporting the *MaxCommittedJitter* output parameter.

2.4.9.3 Control Point requirements when calling the action

If a Control Point (i.e., *QosManager*) sets the *RequestedQosType* field to “0” (Prioritized QoS) or “1” (Hybrid QoS), it MUST supply the *TrafficImportanceNumber* in *AdmitTrafficDescriptor* to *QosDevice* Service when calling the *AdmitTrafficQos()* action.

A Control Point (i.e., *QosManager*) MUST supply the TrafficHandle in *AdmitTrafficDescriptor* to *QosDevice* Service when calling the *AdmitTrafficQos()* action.

A Control Point (i.e., *QosManager*) MUST supply an ActiveTspecIndex that is one of the TspecIndex values in the *AvailableOrderedTspecList* in *AdmitTrafficDescriptor* to *QosDevice* Service when calling the *AdmitTrafficQos()* action.

A Control Point (i.e., *QosManager*) MUST supply the required parameters in the TSPEC (see [QM] for a list of which fields are required).

A Control Point MUST supply a valid *Resource* argument that contains a *DeviceResource* or *NetworkResource* managed by this *QosDevice* Service as indicated in the state variables *A_ARG_TYPE_QosDeviceCapabilities* or *A_ARG_TYPE_QosDeviceExtendedState*.

A Control Point MUST supply a complete *TrafficId* in *AdmitTrafficDescriptor* including *SourceAddress*, *SourcePort*, *DestinationAddress*, *DestinationPort* and *IpProtocol*.

If *AdmitTrafficDescriptor* contains a *TrafficHandle* that is already registered on this *QosDevice* Service for a different Resource, then *AdmitTrafficDescriptor* and the previously registered Traffic Descriptor MUST be equal with the exception of the *QosSegmentSpecificParameters* within the TSPEC field of the Traffic Descriptors.

If the *RequestedQosType* field is set to “1” (Hybrid QoS) or “2” (Parameterized) then a Control Point MUST specify a *TrafficLeaseTime* in the *AdmitTrafficDescriptor* input argument.

If *QosSegmentMaxDelayLow* is specified but *QosSegmentMaxDelayHigh* is not specified then the *QosDevice* Service MUST return error code 711.

2.4.9.4 Dependency on State (if any)

If *AdmitTrafficQos()* or *SetupTrafficQos()* was previously called with the same TrafficHandle (i.e., a currently active TrafficHandle) and an overlapping Resource argument (i.e., the same *InterfaceId*, *LinkId* or *DeviceResourceId* value), this call MUST be treated as an error (error code 717).

2.4.9.5 Effect on State (if any)

As a result of successfully invoking this action, resources are reserved for this stream and the TrafficDescriptor is stored in the *QosDevice* Service.

2.4.9.6 Errors

Table 2-25 — Error Codes for *AdmitTrafficQos()*

errorCode	errorDescription	Description
700	Traffic Handle missing or empty	Traffic Handle MUST be filled in as input to this action.
710	Incomplete TrafficId	All TrafficId fields (SourceAddress, DestinationIp, SourcePort, DestinationPort and IpProtocol) MUST be present.
711	Insufficient information	The input information is not complete.
713	Resource identifier unknown to this device	The indicated DeviceResource or NetworkResource is not managed by this *QosDevice* Service.
717	AdmitTrafficDescriptor already configured	The AdmitTrafficDescriptor as indentified by TrafficHandle is already configured on the indicated Resource.
718	Unknown RequestedQosType	The RequestedQosType in AdmitTrafficDescriptor is present and does not correspond to any known value.
720	ActiveTspecIndex is not a TspecIndex	The ActiveTspecIndex in the AdmitTrafficDescriptor does not correspond to a TspecIndex in the AdmitTrafficDescriptor.
751	Device not on path	This *QosDevice* Service is not on the path of the stream described by the AdmitTrafficDescriptor.
764	Admission Control Not Supported	Admission Control is not supported on this resource (Network or Device Resource)
767	TrafficDescriptors with same TrafficHandle differ	The AdmitTrafficDescriptor's TrafficHandle matches one already registered on this QosDevice and they differ in ways other than QosSegmentSpecificParameters.
791	Missing TrafficLeaseTime	TrafficLeaseTime in TrafficDescriptor is missing for a Parameterized or Hybrid QoS request

2.4.9.7 Reason Codes

Table 2-26 — Reason Codes for *AdmitTrafficQos()*

Reason Code	Reason Description	Description
000	Success	The action succeeded in performing the requested service.
001	Registered	This *QosDevice* Service has determined that another *QosDevice* Service on the QoS Segment is responsible for the request. This *QosDevice* Service has not reserved resources for this stream but has registered the Traffic stream.
762	Insufficient resources	Action failed due to insufficient resources.

2.4.10 UpdateAdmittedQos()

UpdateAdmittedQos() is a required action which is used to update an L2 resource reservation for an existing traffic stream. It may be used to update either a prioritized or a parameterized traffic stream. This action is the recommended over *ReleaseTrafficQos()* followed by *AdmitTrafficQos()* to update an existing stream reservation.

2.4.10.1 Arguments

Table 2-27 — Arguments for *UpdateAdmittedQos()*

Argument	Direction	relatedStateVariable
UpdateTrafficDescriptor	In	*A_ARG_TYPE_TrafficDescriptor*
Resource	In	*A_ARG_TYPE_Resource*
PreemptingTrafficInfo	In	*A_ARG_TYPE_PreemptingTrafficInfo*
AdmitTrafficQosSucceeded	Out	*A_ARG_TYPE_AdmitTrafficQosSucceeded*
AdmitTrafficQosExtendedResult	Out	*A_ARG_TYPE_AdmitTrafficQosExtendedResult*
Layer2MappingContainer	Out	*A_ARG_TYPE_Layer2MappingContainer*

2.4.10.2 Service requirements

If an update fails, the existing reservation MUST remain unaltered.

The *QosDevice* Service MUST preserve the *PreemptingTrafficInfo* state variable and the Traffic Descriptor of the stream being updated so it can report them as part of the *GetUnexpectedStreamChanges()* action if:

- The *QosDevice* Service supports the optional *GetUnexpectedStreamChanges()* action
- And the *ReleaseCausedByPreemption* field in the *PreemptingTrafficInfo* state variable is "**1**"

If the *QosDevice* Service does not receive an *UpdateTrafficDescriptor* with a *TrafficHandle*, or *TrafficHandle* has a NULL value, it MUST return error 700.

If the *QosDevice* Service does not receive an *UpdateTrafficDescriptor* with a *TrafficHandle* that was previously admitted for the indicated *Resource* using *SetupTrafficQos()* or *AdmitTrafficQos()*, it MUST return error 703.

In the *UpdateTrafficDescriptor* to the *QosDevice* Service, the Tspec for which update is requested is indicated by the *ActiveTspecIndex*. *ActiveTspecIndex* MUST be one of the *TspecIndex* values in the *AvailableOrderedTspecList*. If not, *QosDevice* Service MUST return the error 720.

If the *RequestedQosType* field is set to "0" (Prioritized QoS), the *QosDevice* Service MUST update the stream as prioritized.

The *QosDevice* Service MUST return success with *Reason* "001" and MUST NOT update resource reservations on the underlying L2 Technology if:

- The *RequestedQosType* field is set to "2" (Parameterized QoS) or "1" (Hybrid QoS) and the Resource supports parameterized QoS
- And the *QosDevice* Service determines that it is not responsible for reserving resource on that QoS Segment. This is based on the L2 Technology for the Interface and the values of the fields *QDDownstream* and *QDUpstream* in the *Resource* argument.

If the *RequestedQosType* field is set to "1" (Hybrid QoS), the *QosDevice* Service MUST update the stream as parameterized if the QoS Segment supports parameterized QoS; it

MUST update the stream as prioritized if the QoS Segment supports prioritized QoS but does not support parameterized QoS.

If the *RequestedQosType* field is set to “2” (Parameterized QoS), the *QosDevice* Service MUST update the stream as Parameterized. If the request is on a QoS Segment that does not support parameterized QoS then the *QosDevice* Service MUST return error code 764.

If the *RequestedQosType* field is set to a value greater than “2”, the *QosDevice* Service MUST return error 718.

If the stream is updated on a QoS Segment that supports parameterized QoS and *RequestedQosType* field is set to “1” (Hybrid QoS) or “2” (Parameterized QoS), the amount of resources reserved for the stream MUST be sufficient to support the active TSPEC.

If the *QosDevice* Service is able to learn the *Layer2StreamId* of the updated stream, it SHOULD populate the *Layer2MappingContainer* with this information.

If the *Resource* argument contains a *DeviceResource* or *NetworkResource* that is not known to the *QosDevice* Service, it MUST return error 713.

If the *TrafficId* in *UpdateTrafficDescriptor* is incomplete, the *QosDevice* Service MUST return error 710. *TrafficId* includes *SourceAddress*, *SourcePort*, *DestinationAddress*, *DestinationPort* and *IpProtocol*.

If QoS reservation updating is not supported on the L2 Technology on the specified QoS Segment, the *QosDevice* Service MUST return error 763.

If the *RequestedQosType* field is set to “1” (Hybrid QoS) or “2” (Parameterized) and the *UpdateTrafficDescriptor* does not contain a *TrafficLeaseTime*, then the *QosDevice* Service MUST return error 791.

If the underlying L2 Technology is able to report a value for *MaxCommittedDelay* then the *QosDevice* Service that is responsible for reserving resources with the underlying L2 Technology for the requested interface MUST populate the *MaxCommittedDelay* field in the *AdmitTrafficQosExtendedResult* return argument upon successful update when *RequestedQosType* field is set to “1” (Hybrid QoS) or “2” (Parameterized QoS). See clause 2.4.9.2.1 for information on calculating *MaxCommittedDelay.*

If the underlying L2 Technology is able to report a value for *MaxCommittedJitter* then the *QosDevice* Service that is responsible for reserving resources with the underlying L2 Technology for the requested interface MUST populate the *MaxCommittedJitter* field in the *AdmitTrafficQosExtendedResult* return argument upon successful update when *RequestedQosType* field is set to “1” (Hybrid QoS) or “2” (Parameterized QoS).

If *QosSegmentMaxDelayHigh* is provided by the Control Point and *QosSegmentMaxDelayLow* is not provided then *QosDevice* Service MUST provide any delay less than or equal to *QosSegmentMaxDelayHigh* for the segment. If both *QosSegmentMaxDelayHigh* and *QosSegmentMaxDelayLow* are provided then the *QosDevice* Service SHOULD provide the LOWEST possible value greater than or equal to *QosSegmentMaxDelayLow* and less than or equal to *QosSegmentMaxDelayHigh*.

If *UpdateAdmittedQos()* is unable to update the QoS reservation due to lack of resources, it MUST return successful completion with the *Reason* set to “762”.

If *UpdateAdmittedQos()* is unable to update the *UpdateTrafficDescriptor*, it SHOULD populate the *ListOfLayer2StreamIds* in the *AdmitTrafficQosExtendedResults* output argument with all blocking *Layer2StreamIds* on the same QoS Segment. It may not be possible for the *QosDevice* Service to determine all *Layer2StreamIds*.

2.4.10.3 Control Point requirements when calling the action

If a Control Point (i.e., *QosManager*) sets the *RequestedQosType* field to “0” (Prioritized QoS) or “1” (Hybrid QoS), it MUST supply the *TrafficImportanceNumber* in *UpdateTrafficDescriptor* to *QosDevice* Service when calling the *UpdateAdmittedQos()* action.

A Control Point (i.e., *QosManager*) MUST supply the *TrafficHandle* in *UpdateTrafficDescriptor* to *QosDevice* Service when calling the *UpdateAdmittedQos()* action.

A Control Point (i.e., *QosManager*) MUST supply an *ActiveTspecIndex* that is one of the *TspecIndex* values in the *AvailableOrderedTspecList* in *UpdateTrafficDescriptor* to *QosDevice* Service when calling the *UpdateAdmittedQos()* action.

A Control Point (i.e., *QosManager*) MUST supply the required parameters in the TSPEC (see [QM] for a list of which fields are required).

A Control Point MUST supply a valid *Resource* argument that contains a *DeviceResource* or a *NetworkResource* managed by this *QosDevice* Service as indicated in the state variables *A_ARG_TYPE_QosDeviceCapabilities* and *A_ARG_TYPE_QosDeviceExtendedState*.

If a Control Point invokes *UpdateAdmittedQos()* for preemption purposes, it MUST populate *PreemptingTrafficInfo* with the details of the stream that is causing the preemption. In some cases a stream that has been identified as a blocking stream may have additional TSPECs that use fewer resources. The Control Point may choose to free resources used by a blocking stream by updating its QoS reservation to use less resources instead of completely releasing it.

A Control Point MUST supply a complete *TrafficId* in *UpdateTrafficDescriptor* including *SourceAddress*, *SourcePort*, *DestinationAddress*, *DestinationPort* and *IpProtocol*.

If the *RequestedQosType* field is set to “1” (Hybrid QoS) or “2” (Parameterized) then a Control Point MUST specify a *TrafficLeaseTime* in the *UpdateTrafficDescriptor* input argument.

If *QosSegmentMaxDelayLow* is specified but *QosSegmentMaxDelayHigh* is not specified then the *QosDevice* Service MUST return error code 711.

2.4.10.4 Dependency on State (if any)

The *TrafficHandle* passed in as an argument MUST have been successfully admitted by *AdmitTrafficQos()* for the *Resource*. Otherwise it MUST be treated as an error (error code 703).

2.4.10.5 Effect on State (if any)

If the *QosDevice* Service supports the optional *GetUnexpectedStreamChanges()* action the *QosDevice* Service preserves the information contained in *PreemptingTrafficInfo* argument (see *AffectingStreamTrafficDescriptor* in clause 2.2.22.3) and the Traffic Descriptor of the updated stream (see *AffectedStreamTrafficDescriptor* in clause 2.2.22.3).

2.4.10.6 Errors

Table 2-28 — Error Codes for *UpdateAdmittedQos()*

errorCode	errorDescription	Description
700	Traffic Handle missing or empty	Traffic Handle MUST be filled in as input to this action.
703	Traffic Handle unknown to this device	Traffic Handle (i.e., Traffic Descriptor) is unknown to this device.
710	Incomplete TrafficId	All TrafficId fields (SourceAddress, DestinationIp, Source Port, Destination Port and IpProtocol) MUST be present.
711	Insufficient information	The input information is not complete.
713	Resource identifier unknown to this device	The indicated DeviceResource or NetworkResource is not managed by this *QosDevice* Service.
718	Unknown RequestedQosType	The RequestedQosType in AdmitTrafficDescriptor is present and does not correspond to any known value.
720	ActiveTspecIndex is not a TspecIndex	The ActiveTspecIndex does not match the TspecIndex of any of the TSPECs contained within the Traffic Descriptor.
763	Update not supported	Action failed because updating of reserved resources is not supported on this QosSegment
764	Admission Control Not Supported	Admission Control is not supported on this resource (Network or Device Resource)
791	Missing TrafficLeaseTime	TrafficLeaseTime in TrafficDescriptor is missing for a Parameterized or Hybrid QoS request

2.4.10.7 Reason Codes

Table 2-29 — Reason Codes for *UpdateAdmittedQos()*

Reason Code	Reason Description	Description
000	Success	The action succeeded in performing the requested service.
001	Registered	This *QosDevice* Service has determined that another *QosDevice* Service on the QosSegment is responsible for the request. This *QosDevice* Service has not reserved resources for this stream but has registered the Traffic stream.
762	Insufficient resources	Action failed due to insufficient resources.

2.4.11 ReleaseAdmittedQos()

ReleaseAdmittedQos() provides an action to direct the *QosDevice* Service to release the L2 QoS resources for the traffic identified by *ReleaseTrafficHandle* and *Resource*. Only the resources identified by the *TrafficHandle* and *Resource* arguments are released.

2.4.11.1 Arguments

Table 2-30 — Arguments for *ReleaseAdmittedQos()*

Argument	Direction	relatedStateVariable
ReleaseTrafficHandle	In	*A_ARG_TYPE_TrafficHandle*
Resource	In	*A_ARG_TYPE_Resource*
PreemptingTrafficInfo	In	*A_ARG_TYPE_PreemptingTrafficInfo*

2.4.11.2 Service requirements

The *QosDevice* Service MUST preserve the *PreemptingTrafficInfo* state variable and the Traffic Descriptor of the Stream being updated so it can report them as part of the *GetUnexpectedStreamChanges()* action if:

- The *QosDevice* Service supports the optional *GetUnexpectedStreamChanges()* action
- And the *ReleaseCausedByPreemption* field in the *PreemptingTrafficInfo* state variable is "1"

If the *QosDevice* Service does not receive a *TrafficHandle*, or *TrafficHandle* has a NULL value, it MUST return error 700.

If the *Resource* argument contains a valid *DeviceResource* or *NetworkResource* that is managed by this *QosDevice* Service, it MUST attempt to release resources for the indicated *Resource* only.

If the *Resource* argument contains a *DeviceResource* or *NetworkResource* that is not known to the QosDevice Service, it MUST return error 713.

If the *QosDevice* Service does not receive a *TrafficHandle* that was previously admitted for the indicated *Resource* using *AdmitTrafficQos()*, it MUST return error 703.

2.4.11.3 Control Point requirements when calling the action

A Control Point (i.e., QosManager) MUST supply the *TrafficHandle* to *QosDevice* Service when calling the *ReleaseAdmittedQos()* action.

A Control Point MUST supply a valid *Resource* argument that contains a *DeviceResource* or *NetworkResource* managed by this *QosDevice* Service as indicated in the state variables *A_ARG_TYPE_QosDeviceCapabilities* and *A_ARG_TYPE_QosDeviceExtendedState*.

If a Control Point invokes *ReleaseAdmittedQos()* for preemption purposes, it MUST populate *PreemptingTrafficInfo* with the details of the stream that is causing the preemption.

2.4.11.4 Dependency on State (if any)

The *TrafficHandle* passed in as an argument MUST have been successfully admitted by *SetupTrafficQos()* or, preferably, *AdmitTrafficQos()* for the *Resource*. Otherwise it MUST be treated as an error (error code 703).

2.4.11.5 Effect on State (if any)

If the *QosDevice* Service supports the optional *GetUnexpectedStreamChanges()* action the *QosDevice* Service preserves the information contained in *PreemptingTrafficInfo* argument (see *AffectingStreamTrafficDescriptor* in clause 2.2.22.3) and the Traffic Descriptor of the released stream (see *AffectedStreamTrafficDescriptor* in clause 2.2.22.3).

Information related to this stream need not be kept any longer than needed to meet the requirements for the *GetUnexpectedStreamChanges()* action, if applicable.

2.4.11.6 Errors

Table 2-31 — Error Codes for *ReleaseAdmittedQos()*

errorCode	errorDescription	Description
700	Traffic Handle missing or empty	Traffic Handle MUST be filled in as input to this action.
703	Traffic Handle unknown to this device	The TrafficHandle is unknown to this device
713	Resource identifier unknown to this device	The indicated DeviceResource or NetworkResource is not managed by this *QosDevice* Service.

2.4.12 GetExtendedQosState()

GetExtendedQosState() provides information about the capabilities and current state of the *QosDevice* Service. If a *TrafficDescriptor* is provided as an input argument, the *QosDevice* Service MAY also provide information on the TSPECs that it can support for the given *TrafficDescriptor* and MAY provide *TrafficId* information as well.

This action combines and extends the functionality of the *GetQosCapabilities(), GetQosState()* and *GetQosDeviceInfo()* actions. Use of the *GetExtendedQosState()* action is recommended.

2.4.12.1 Arguments

Table 2-32 — Arguments for *GetExtendedQosState()*

Argument	Direction	relatedStateVariable
InputTrafficDescriptor	In	*A_ARG_TYPE_TrafficDescriptorContainer*
TrafficDescriptorsWanted	In	*A_ARG_TYPE_TrafficDescriptorsWanted*
QosDeviceExtendedState	Out	*A_ARG_TYPE_QosDeviceExtendedState*
QosDeviceInfo	Out	*A_ARG_TYPE_QosDeviceInfoContainer*
ListOfAdmittedTraffic	Out	*A_ARG_TYPE_ListOfAdmittedTraffic*

2.4.12.2 Service requirements

If *InputTrafficDescriptor* argument contains a *TrafficDescriptor*, the *QosDevice* Service SHOULD attempt to populate the *ListOfProtoTspecs* contained in *QosDeviceExtendedState* with information that describes the resources that may be available when attempting to admit that *TrafficDescriptor*.

If *InputTrafficDescriptor* argument contains a *TrafficDescriptor*, the *QosDevice* Service SHOULD attempt to populate the *QosDeviceInfo* output argument.

If *InputTrafficDescriptor* argument contains a *TrafficDescriptor* and that *TrafficDescriptor* does not contain a *TrafficHandle* registered with this *QosDevice* Service, the *QosDevice* Service MUST return error code 700.

If *TrafficDescriptorsWanted* is set to "**1**", the *QosDevice* Service MUST populate the *ListOfAdmittedTraffic* with the TrafficDescriptors managed by it.

The *QosDevice* Service MUST populate the required fields needed to describe the state and capabilities of this *QosDevice* Service in the *QosDeviceExtendedState* output argument.

If a TrafficDescriptor is provided in the input argument *InputTrafficDescriptor* which is insufficient to determine the port numbers and protocol the *QosDevice* Service will not populate *QosDeviceInfoContainer*.

2.4.12.3 Control Point requirements when calling the action

In order to get ProtoTspecs a Control Point (i.e., *QosManager*) calling the action *GetExtendedQosState()* MUST supply a *TrafficDescriptor* by passing it within the *InputTrafficDescriptor* argument. Otherwise, *InputTrafficDescriptor* MUST be left empty.

If a Control Point wants the list of TrafficDescriptors known to the *QosDevice* Service to be returned in the *ListOfAdmittedTraffic* output argument, it MUST pass in a value of "**1**" for the *TrafficDescriptorsWanted* argument.

2.4.12.4 Dependency on State (if any)

None

2.4.12.5 Effect on State (if any)

None

2.4.12.6 Errors

Table 2-33 — Error Codes for *GetExtendedQosState()*

errorCode	errorDescription	Description
700	Unknown *TrafficHandle*	*InputTrafficDescriptor* contains a *TrafficHandle* that is not registered with this *QosDevice* Service.

2.4.13 SetPreferredQph()

This is an optional action. This action enables the feature of the preferred *QosPolicyHolder*. It provides storage on this device of the identity of the preferred *QosPolicyHolder*. See [QPH]for more details.

2.4.13.1 Arguments

Table 2-34 — Arguments for *SetPreferredQph()*

Argument	Direction	relatedStateVariable
PreferredQph	In	*A_ARG_TYPE_PreferredQph*
CurrentPreferredQph	Out	*A_ARG_TYPE_PreferredQph*
SetPreferredQphResults	Out	*A_ARG_TYPE_SetPreferredQphResults*

2.4.13.2 Service requirements

To query the currently preferred QPH information without changing the value of the preferred QPH, the *QphPreferenceCount* field in the *PreferredQph* input argument is set to "0". If the number in the *QphPreferenceCount* field in the *PreferredQph* input argument is "0", the *QosDevice* Service MUST return the currently stored value of the *PreferredQphId* and *QphPreferenceCount* as part of the *CurrentPreferredQph* output argument. The default currently stored value (i.e., currently stored value before the first successful invocation of the *SetPreferredQph*() action) for *PreferredQphId* field MUST be NULL and for *QphPreferenceCount* field MUST be 0. The *QosDevice* Service MUST return Successful completion with *SetPreferredQphResults* set to "0" to indicate that the action was successful.

If the *QphPreferenceCount* field in the *PreferredQph* is less than the currently stored value of the *PreferredQphId*, the *QosDevice* Service MUST return the currently stored value of the

PreferredQphId and *QphPreferenceCount* as part of the *CurrentPreferredQph* output argument. The *QosDevice* Service MUST return Successful completion with *SetPreferredQphResults* “770” to indicate that this is not the latest preferred QosPolicyHolder Service. This value for *SetPreferredQphResults* serves to inform the invoking QosPolicyHolder Service that it is no longer preferred.

If the number in the *QphPreferenceCount* field in the *PreferredQph* is equal to the currently stored value of the *PreferredQphId* and the *PreferredQphId* is the same as the currently stored value of the *PreferredQphId*, the *QosDevice* Service MUST return the currently stored value of the *PreferredQphId* and *QphPreferenceCount* as part of the *CurrentPreferredQph* output argument. The *QosDevice* Service MUST return Successful completion with *SetPreferredQphResults* set to “0” to indicate that the action was successful.

If the number in the *QphPreferenceCount* field in the *PreferredQph* is equal to the currently stored value of the *PreferredQphId* and the *PreferredQphId* is not the same as the currently stored value of the *PreferredQphId*, the *QosDevice* Service MUST return the currently stored value of the *PreferredQphId* and *QphPreferenceCount* as part of the *CurrentPreferredQph* output argument. The *QosDevice* Service MUST return Successful completion with *SetPreferredQphResults* set to “771” to indicate that a synchronization error occurred.

If the number in the *QphPreferenceCount* field in the *PreferredQph* is larger than the currently stored value of the *PreferredQphId*, the *QosDevice* Service MUST store the value of the *QphPreferenceCount* and the value of the *PreferredQphId* from the input argument. The *QosDevice* Service MUST return the newly stored values for *QphPreferenceCount* and *PreferredQphId* as part of the *CurrentPreferredQph* output argument. The *QosDevice* Service MUST return Successful completion with *SetPreferredQphResults* set to “0” to indicate that the action was succesful.

If the *SetPreferredQph()* action succeeds in setting a new Preferred QosPolicyHolder, the *QosDevice* Service MUST save the information contained in the *PreferredQph* argument across reboots and power cycling (e.g., store it in non-volatile memory).

2.4.13.3 Control Point requirements when calling the action

A Control Point (i.e., QosPolicyHolder Service) calling action *SetPreferredQph()* MUST provide a valid QosPolicyHolder ID in the *PreferredQphId* argument.

2.4.13.4 Dependency on State (if any)

The dependencies on state are not through an explicit state variable. Only requests with a higher *QphPreferenceCount* than the one stored in the *QosDevice* Service will be able to succeed.

2.4.13.5 Effect on State (if any)

Successfully invoking this *SetPreferredQph()* with a larger *QphPreferenceCount* than the one that was stored previously changes the preferred *QosPolicyHolder* Service known to this *QosDevice* Service.

2.4.13.6 Errors

Table 2-35 — *SetPreferredQphResults* for *SetPreferredQph()*

SetPreferredQphResults	Result Description	Description
770	QphPreferenceCount mismatch	The QphPreferenceCount in PreferredQphId passed in to SetPreferredQph is lower than that stored in the *QosDevice* Service.
771	Synchronization error	A synchronization error occurred.

2.4.14 GetUnexpectedStreamChanges()

This optional action is called by a Control Point to learn the TrafficDescriptors of the streams that were recently released and/or updated due to preemption and the TrafficDescriptors of the streams that caused the changes. If this action is implemented, there is no required minimum number of unexpected stream changes which should be kept. It is recommended that the *QosDevice* Service keep information about all unexpected stream changes since the last *UnexpectedStreamChange* event, or that occurred in at least the last sixty seconds, whichever is greater. The *UnexpectedStreamChange* state variable MUST be synchronized with the *UnexpectedStreamChangeIndex*.

2.4.14.1 Arguments

Table 2-36 — Arguments for *GetUnexpectedStreamChanges()*

Argument	Direction	relatedStateVariable
NumberOfUnexpectedStreamChangesRequested	In	*A_ARG_TYPE_NumberOfUnexpectedStreamChangesRequested*
NumberOfUnexpectedStreamChangesReported	Out	*A_ARG_TYPE_NumberOfUnexpectedStreamChangesReported*
MostRecentUnexpectedStreamChanges	Out	*A_ARG_TYPE_ListOfMostRecentUnexpectedStreamChanges*

2.4.14.2 Service requirements

The *QosDevice* Service MUST populate *NumberOfUnexpectedStreamChangesReported* with the number of *UnexpectedStreamChanges* in the output argument *MostRecentUnexpectedStreamChanges*. The *NumberOfUnexpectedStreamChangesReported* MUST NOT exceed *NumberOfUnexpectedStreamChangesRequested*.

How many unexpected stream changes are kept and for how long they are kept is implementation-specific. An indication MUST also be preserved that this was a release of the stream rather than an update of the reservation in the *MostRecentUnexpectedStreamChanges* output argument.

2.4.14.3 Control Point requirements when calling the action

A Control Point (i.e., *QosManager*) calling action *GetUnexpectedStreamChanges()* MUST pass in a value for *NumberOfUnexpectedStreamChangesRequested* that indicates maximum number of *UnexpectedStreamChanges* to be placed in *MostRecentUnexpectStrreamChanges*.

2.4.14.4 Dependency on State (if any)

None

2.4.14.5 Effect on State (if any)

None

2.4.14.6 Errors

Table 2-37 — Error Codes for *GetUnexpectedStreamChanges()*

errorCode	errorDescription	Description

2.4.15 VerifyTrafficHandle()

The *VerifyTrafficHandle()* action is an optional action that is used to verify that a TrafficHandle is valid on this *QosDevice* Service. If the requested TrafficHandle is registered on this *QosDevice* Service, it returns the TrafficDescriptor associated with the TrafficHandle. This action enables a Control Point to verify whether a given stream is reserving resources.

2.4.15.1 Arguments

Table 2-38 — Arguments for *VerifyTrafficHandle()*

Argument	Direction	relatedStateVariable
TrafficHandleToVerify	In	*A_ARG_TYPE_TrafficHandle*
TrafficDescriptorWanted	In	*A_ARG_TYPE_TrafficDescriptorsWanted*
TrafficDescriptor	Out	*A_ARG_TYPE_TrafficDescriptorContainer*

2.4.15.2 Service requirements

The *QosDevice* Service MUST return an error code of 700 if the input TrafficHandle is empty or NULL. It MUST return an error code of 703 if the TrafficHandle is not managed by this *QosDevice* Service.

If *TrafficDescriptorWanted* is "1", the *QosDevice* Service MUST populate the *TrafficDescriptor* output argument.

If the *TrafficHandleToVerify* input argument identifies a TrafficDescriptor which is registered on multiple Resources, then the *TrafficDescriptor* output argument MUST contain a *QosSegmentSpecificParameters* structure for each Resource on which it is registered.

2.4.15.3 Control Point requirements when calling the action

None

2.4.15.4 Dependency on State (if any)

None

2.4.15.5 Effect on State (if any)

None.

2.4.15.6 Errors

Table 2-39 — Error Codes for *VerifyTrafficHandle()*

errorCode	errorDescription	Description
700	Traffic Handle missing or empty	Traffic Handle MUST be filled in as input to this action.
703	Traffic Handle unknown to this device	The TrafficHandle is unknown to this device or is no longer associated with an active L2 Stream

2.4.16 UpdateTrafficLeaseTime()

The *UpdateTrafficLeaseTime()* action provides a method to extend the lease time of a given stream. This is a required action.

2.4.16.1 Arguments

Table 2-40 — Arguments for *UpdateTrafficLeaseTime()*

Argument	Direction	relatedStateVariable
TrafficHandle	In	*A_ARG_TYPE_TrafficHandle*
NewTrafficLeaseTime	In	*A_ARG_TYPE_NewTrafficLeaseTime*

2.4.16.2 Service requirements

The *QosDevice* Service MUST return an error code of 700 if the input *TrafficHandle* is empty or NULL. The *QosDevice* Service MUST return an error code 703 if the input *TrafficHandle* is not managed by this *QosDevice* Service.

The *QosDevice* Service MUST update the lease times of all of its reservations associated with that *TrafficHandle*.

2.4.16.3 Control Point requirements when calling the action

Control Point MUST supply a *TrafficHandle* managed by this *QosDevice* Service in order to update the lease time of the QoS for the indicated stream.

2.4.16.4 Dependency on State (if any)

The *TrafficHandle* provided has to be managed by the *QosDevice* Service.

2.4.16.5 Effect on State (if any)

After this call, the lease time for the indicated stream is updated.

2.4.16.6 Errors

Table 2-41 — Error Codes for *UpdateTrafficLeaseTime()*

errorCode	errorDescription	Description
700	Traffic Handle missing or empty	Traffic Handle MUST be filled in as input to this action.
703	Traffic Handle unknown to this device	The TrafficHandle is not managed by this device or is no longer associated with an active L2 Stream

2.4.17 SetL2Map()

The *SetL2Map()* action provides the Control Point (i.e., *QosManager*) with the ability to convey a *Layer2StreamId* for a registered TrafficDescriptor to a *QosDevice* Service that does not already know the *Layer2StreamId*. If the TrafficDescriptor already has a *Layer2StreamId* associated with it, *SetL2Map()* will update this value.

This action is optional because some L2 Technologies may not support *Layer2StreamId* whereas other L2 Technologies provide L2 mechanisms that provide the same results.

After invocation of this action the *QosDevice* Service can identify the UPnP-QoS TrafficHandle with the *Layer2StreamId*. This allows the *QosDevice* Service to determine which packets belong to the traffic stream managed by the L2 Technology.

2.4.17.1 Arguments

Table 2-42 — Arguments for *SetL2Map()*

Argument	Direction	relatedStateVariable
Layer2Mapping	In	*A_ARG_TYPE_Layer2Mapping*

2.4.17.2 Service requirements

The *QosDevice* Service MUST return an error code 703 if the *TrafficHandle* field in the *Layer2Mapping* input argument is not a known to this *QosDevice* Service.

The *QosDevice* Service MUST return an error code 700 if the *TrafficHandle* field in the *Layer2Mapping* input argument is empty.

2.4.17.3 Control Point requirements when calling the action

Control Point MUST supply a valid *TrafficHandle* to set the *Layer2StreamId* of the traffic stream.

2.4.17.4 Dependency on State (if any)

The *TrafficHandle* provided has to be known to the *QosDevice* Service.

2.4.17.5 Effect on State (if any)

After this call, the *QosDevice* Service will know the *Layer2StreamId* associated with the traffic stream.

2.4.17.6 Errors

Table 2-43 — Error Codes for *SetL2Map()*

errorCode	errorDescription	Description
700	TrafficHandle missing or empty	TrafficHandle is empty or missing.
703	TrafficHandle unknown to this device	The TrafficHandle field in the Layer2Mapping input argument is not known to this *QosDevice* Service.

2.4.18 Non-Standard Actions Implemented by a UPnP Vendor

To facilitate certification, non-standard actions implemented by UPnP vendors should be included in this service template. The UPnP Device Architecture lists naming requirements for non-standard actions (see the clause on Description).

2.4.19 Error Code Summary

The following table lists the error codes used in actions for this service type. If an action results in multiple errors, the most specific error MUST be returned. The common error codes are defined in the UPnP Device Architecture [DEVICE].

Table 2-44 — Error Code Summary

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture clause on Control. [DEVICE]
500-599	TBD	See UPnPDevice Architecture clause on Control. [DEVICE]
600-699	TBD	See UPnP Device Architecture clause on Control. [DEVICE]

errorCode	errorDescription	Description
700	Traffic Handle missing or empty	Traffic Handle is missing or empty.
702	Traffic Handle already registered	A Control Point (i.e., QosManager) is not allowed to set up or modify QoS using *SetupTrafficQos()* if QoS has already been set up for that handle.
703	Traffic Handle unknown to this device	The TrafficHandle is not managed by this device or is no longer associated with an active L2 Stream
710	Incomplete TrafficId	All TrafficId fields (Source Ip, Destination IP, Source Port, Destination Port and IpProtocol) MUST be present.
711	Insufficient information	The input information is not complete.
712	Incomplete information to determine protocol and port numbers	Incomplete information. For example, in case of the UPnP AV scenario, MediaServerConnectionId, MediaRendererConnectionId, AvTransportUri or AvTransportInstanceId is required but not provided.
713	Resource Identifier unknown to this device	Incomplete information. For example, in case of the UPnP AV scenario, MediaServerConnectionId, MediaRendererConnectionId, AvTransportUri or AvTransportInstanceId is required but not provided.
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
717	Illegal action on this TrafficHandle	An action that is illegal for this TrafficHandle has been invoked. E.g., *ReleaseTrafficQos()* invoked for a parameterized stream.
720	ActiveTspecIndex is not a TspecIndex	The ActiveTspecIndex does not correspond to the TspecIndex of any of the TSPECs contained in the TrafficDescriptor.
730	ROPeriod incapable	ROPeriod is outside the capabilities of the device.
731	MonitorResolutionPeriod incapable	MonitorResolutionPeriod is outside the capabilities of the device.
732	Requested too many observations	RequestedNumRotameterObservations is more than device capabilities.
733	No valid observation	Unable to provide an observation at this time.
734	Invalid arguments	ROPeriod cannot be greater than MonitorResolutionPeriod
735	*ConfigureRotameterObservation()* has not been invoked	*ConfigureRotameterObservation()* has not been invoked before calling GetRotameterObservation
751	Device not on path	This *QosDevice* Service is not on the stream's path.
760	QosStateId does not match	Please refer to the 'Theory of Operation' clause.
761	*QosDevice* Service cannot set up this stream	QoS Setup failed, e.g. device does not support prioritized QoS
763	Update not supported	Action failed because updating of reserved resources is not supported on this QosSegment
791	Missing TrafficLeaseTime	TrafficLeaseTime in TrafficDescriptor is missing for a Parameterized or Hybrid QoS request
800-899	TBD	(Specified by UPnP™ vendor.)

2.4.20 Reason Code Summary

The following table lists the reason codes used in actions for this service type. Reason codes are used in place of error codes when it is necessary for the action to return arguments

describing the cause of the error. The actions for which this is the case are *AdmitTrafficQos()*, *UpdateAdmittedQos()* and *SetPreferedQph()*.

Table 2-45 — Common Reason Codes

ReasonCode	Reason Description	Description
000	Success	The action succeeded in performing the requested service.
001	Registered	This *QosDevice* Service has determined that another *QosDevice* Service on the QosSegment is responsible for the request. This *QosDevice* Service is remaining passive but has registered the Traffic stream and is aware of it.
762	Insufficient resources	Action failed due to insufficient resources.
764	Admission Control Not Supported	Admission Control is not supported on this resource (Network or Device Resource)
770	PreferredQph Failure	The requested (input) PreferredQph cannot be set as the preferred QosPolicyHolder Service
771	SyncError	A synchronization error has occurred. E.g., the PreferredQphCount is the one currently used but it is associated with a different PreferredQphId
800-899	TBD	(Specified by UPnP™ vendor.)

2.5 Theory of Operation (Informative)

The changes made to the *QosDevice* Service to support Parameterized QoS in version 3 are extensive. It has been necessary to extend the functionality of many of the actions present in earlier versions of the *QosDevice* Service with new actions in order to support the new functionality. All of the original actions have been preserved for compatibility with older versions of other UPnP-QoS services and control points but their use is discouraged for UPnP-QoS version 3. For simplicity of description, the Theory of Operation has been separated into two separate subclauses, clause 2.5.1 for the new *QosDevice* Service supporting Parameterized QoS functionality and clause 2.5.2 for Prioritized QoS and backwards compatible functionality.

The *QosDevice* Service provides an interface for Control Points to query the QoS state of the device, perform QoS-related operations on the device and register for events that the device generates. The *QosManager*[QM] interacts with the *QosDevice* Service. While the actions and events are available to any control point, all of them are described herein as being used by the *QosManager*[QM] or *QosPolicyHolder* Service[QPH].

Table 2-46 lists all of the actions defined for the *QosDevice* Service and their use by control points of various versions. Please note that the actions listed as discouraged in version 3 are implemented in a compliant *QosDevice* Service v3; they may be called by legacy control points. While these discouraged actions may be called by version 3 Control Points they will not provide any v3 functionality not already present in v2 and may not provide the complete information needed for other v3 actions. Please note also that the results of the v3 actions are identical to the results of the discouraged actions when invoked for a prioritized traffic stream.

Table 2-46 — Actions in Version 3 and Version 2

Name	Version 3 Req. or Opt. or Discouraged [a]	Version 2 Req. or Opt. [a]
GetQosDeviceCapabilities()	D (Use *GetExtendedQosState()*)	R
GetQosState()	D (Use *GetExtendedQosState()*)	R
SetupTrafficQos()	*D (Use AdmitTrafficQos())*	R
ReleaseTrafficQos()	*D (Use ReleaseAdmittedQos())*	R
GetPathInformation()	*R*	O
GetQosDeviceInfo()	O and D (Use *GetExtendedQosState()*)	O
GetRotameterInformation()	O	O
ConfigureRotameterObservation()	O	O
AdmitTrafficQos()	R	X
UpdateAdmittedQos()	R	X
ReleaseAdmittedQos()	R	X
GetExtendedQosState()	R	X
SetPreferredQph()	O	X
GetUnexpectedStreamChanges()	O	X
VerifyTrafficHandle()	O	X
UpdateTrafficLeaseTime()	R	X
SetL2Map()	O	X
[a] R = Required, O = Optional, D = Discouraged, X = Non-standard		

Table 2-47 lists all the state variables defined for *QosDevice* Service v3 and whether each state variable is Required, Optional, Discouraged or Non-standard for both v2 and v3. The *QosDevice* Service is required to support a state variable if it is listed as required in either v2 or v3.

If a state variable is Discouraged for v3 then the *QosDevice* Service returns values in that state variable that are suitable for v2 *QosManagers.*

Table 2-47 — State Variables in Version 3 and Version 2

Variable Name	Version 3 Req. or Opt. [a]	Version 2 Req. or Opt. [a]
A_ARG_TYPE_TrafficDescriptor	R	R
A_ARG_TYPE_TrafficDescriptorsPerInterface	R	R
A_ARG_TYPE_TrafficHandle	R	R
A_ARG_TYPE_NumTrafficDescriptors	R	R
A_ARG_TYPE_QosDeviceCapabilities	D	R
A_ARG_TYPE_QosDeviceState	D	R
PathInformation	R	O
A_ARG_TYPE_QosDeviceInfo	R	O
A_ARG_TYPE_QosStateId	D	R
A_ARG_TYPE_NumRotameterObservations	O	O
A_ARG_TYPE_RotameterInformation	O	O
A_ARG_TYPE_ConfRotameterObservations	O	O
MostRecentStreamAction	O	O
A_ARG_TYPE_MaxPossibleRotameterObservations	O	O
A_ARG_TYPE_Resource	R	X
A_ARG_TYPE_AdmitTrafficQosExtendedResult	R	X
A_ARG_TYPE_ListOfAdmittedTraffic	R	X
A_ARG_TYPE_PreferredQph	O	X
UnexpectedStreamChange	O	X
A_ARG_TYPE_PreemptingTrafficInfo	O	X
A_ARG_TYPE_ListOfMostRecentUnexpectedStreamChanges	O	X
A_ARG_TYPE_QosDeviceExtendedState	R	X
A_ARG_TYPE_Layer2Mapping	R	X
A_ARG_TYPE_AdmitTrafficQosSucceeded	R	X
A_ARG_TYPE_TrafficDescriptorsWanted	R	X
A_ARG_TYPE_SetPreferredQphResults	O	X
A_ARG_TYPE_NumberOfUnexpectedStreamChangesRequested	O	X
A_ARG_TYPE_NumberOfUnexpectedStreamChangesReported	O	X
A_ARG_TYPE_TrafficDescriptorContainer	R	X
A_ARG_TYPE_Layer2MappingContainer	R	X
A_ARG_TYPE_QosDeviceInfoContainer	R	X
A_ARG_TYPE_NewTrafficLeaseTime	R	X
[a] R = Required, O = Optional, D = Discouraged, X = Non-standard		

2.5.1 Parameterized QoS

A UPnP *QosDevice* Service exposes its QoS capabilities and its current QoS state through the *GetExtendedQosState()* action. This action returns both its static properties (i.e., capabilities and states that are independent of the state of the device) and its dynamic properties (i.e., capabilities and states that depend upon the state of the device at the given time the action is invoked). The information returned by this action provides the *QosManager* with a complete representation of the characteristics of the *QosDevice* Service. Based on this, the *QosManager* determines what steps to take to set up the QoS for new traffic streams. Optionally, the *QosManager* provides a *TrafficDescriptor* as an input argument to the *GetExtendedQosState()* action. If provided, the *QosDevice* Service returns a set of

ProtoTspecs, each of which consist of a subset of TSPEC parameters each with a value or range of values which it can support for the given device. Additionally, if a *TrafficDescriptor* input argument is provided, the *QosDevice* Service may return *QosDeviceInfo* information.

The majority of the information provided by the *GetExtendedQosState()* action pertains to network properties. The properties are contained within three different containers which are described as follows:

- **DeviceResources**: contains properties that are associated with the device's non-networking capabilities (e.g., buffers).
- **Interface**: contains properties that are associated with a specific network interface.
- **Link**: contains properties that are associated with a link. Link containers are always embedded within an interface container.

The QoS related properties of the *QosDevice* Service are expressed in a hierarchical form. The hierarchy is represented by containers. Each container defines the context of the contained properties. There are multiple container types; each type is specific to the device and network aspect. Containers are nested to match the characteristics of the device and network that they represent.

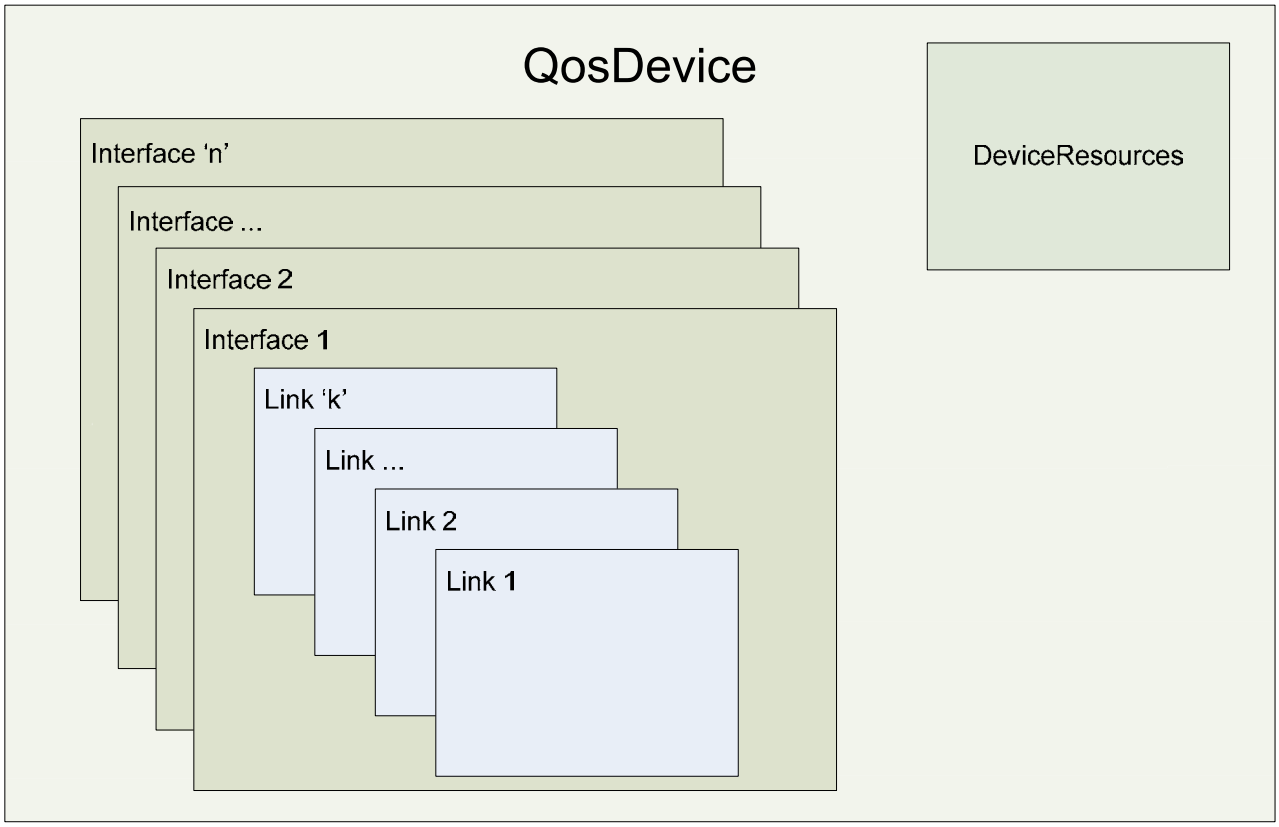


Figure 2-7 — Containers and How They Nest

A given *QosDevice* Service could have more than one Interface to a single L2 Technology. An example of multiple Interfaces is a PC with two Wireless adapters.

Not every Interface contains Links. Links only exist when there are two or more "paths" within an L2 Technology that have different endpoints that could be controlled separately. Different Links on the same Interface may exhibit different QoS-related properties. An example would

be an 802.11 network where a *QosDevice* Service on a station which would have one link container representing the Link between the access point (AP) and the *QosDevice* Service. There could also be additional link containers corresponding to any Direct Link Setup [IEEE11] connections that exist between this *QosDevice* Service and other stations.

The *GetExtendedQosState()* action accepts a *TrafficDescriptor* as an input argument. If a *TrafficDescriptor* is provided as an input argument, the *QosDevice* Service returns information to support the *QosManager* in determining the delay constraints that exist on the QoS Segment so the *QosManager* can apportion the end-to-end delay from the traffic descriptor across the QoS Segments on the path.

The *AdmitTrafficQos()* action provides a mechanism for the *QosManager* to request the *QosDevice* Service to provide network resources or device resources for a traffic stream identified by the *TrafficDescriptor*. Both parameterized and prioritized QoS may be established by this action. The *QosDevice* Service determines which type of QoS to establish by inspecting the *RequestedQosType* parameter within the Active TSPEC.

For parameterized QoS requests the *AdmitTrafficQos()* action is invoked for each interface on each *QosDevice:3* Service on the path. The path of the traffic stream is determined using the *QosDeviceExtendedState* information and the *PathInformation*. The *InterfaceId* (and if necessary the *LinkId*) obtained from the *QosDeviceExtendedState* structure is used when invoking the *AdmitTrafficQos()* action. For *QosDevice* Services in the middle of the path that manage multiple interfaces on the path (i.e., bridges), the *AdmitTrafficQos()* action is invoked separately for each interface.

The *QosDevice* Service will perform the technology specific actions needed to admit the traffic when *AdmitTrafficQos()* action is invoked. Results are returned indicating whether or not the traffic stream was admitted. If the traffic stream was not admitted additional guidance (i.e., a list of blocking streams) may be provided that can be used in future requests.

A separate *UpdateAdmittedQos()* action is provided to permit the modification of already admitted streams.

When an admission request or an update request is refused the *QosManager* or Control Point may, if requested by the Control Point, attempt to free resources on the *QosDevice* Service necessary to allow the new traffic stream. The *QosDevice* Service will provide a list of the streams in the segment that are blocking the new stream. From this list the *QosManager* or Control Point determines which of the blocking streams are of lower importance than the new one by consulting with the *QosPolicyHolder* and thus are candidates to free up network resources.

Even if *AdmitTrafficQos()* returns success, QoS may not have been established for the stream. Since UPnP mandates that no arguments are returned in the case where an action error is returned, the *Reason* field was created to allow *AdmitTrafficQos()* to return useful information when the admission request is denied. When the *AdmitTrafficQos()* action returns success (i.e., HTTP/1.1 200 OK), the *Reason* field in *AdmitTrafficQosExtendedResults* contains information indicating whether *AdmitTrafficQos()* was able to establish QoS or not. Control Points that receive success from the action check the *Reason* field to determine if admission was successful. If admission was not successful, additional information such as a list of blocking streams may be provided in the *AdmitTrafficQosExtendedResults* output argument.

The *QosManager* can indicate to the *QosDevice* Service to release resources in two ways. The *QosManager* may either call the *ReleaseAdmittedQos()* action or simply not renew the lease time associated with the traffic stream. This latter approach is not recommended.. QoS established with *AdmitTrafficQos()* can be correctly released using *ReleaseTrafficQos()*,but is not recommended. In this case *ReleaseAdmittedQos()* is recommended instead.

The *QosDevice* Service provides path and QoS Segment information to the *QosManager* using the *GetPathInformation()* action. This information includes the link identifier, MAC address, bridging information and reachable MAC addresses for each link on the *QosDevice* Service. This is essential to allow a *QosManager* determine the topology of the network and the path of the traffic streams.

Determination of the *QosDevice* Services that are attached to the same QoS Segment is made through the use of the *QosSegmentId*. If the *QosSegmentId* matches, the *QosDevice* Services are attached to the same QoS Segment.

The *QosManager* is not expected to know which *QosDevice* Service on a given QoS Segment is required to reserve resources for the QoS Segment. The *QosManager* invokes the *AdmitTrafficQos()* action on every *QosDevice* Service:3 on the path of the traffic stream. It is up to the *QosDevice* Services to determine which of them are responsible for reserving the L2 Technology resources on a given QoS Segment. In the *Resource* argument of the *AdmitTrafficQos()* and *UpdateAdmittedQos()* actions, the *QDUpstream* and *QDDownstream* fields inform the *QosDevice* Service of any upstream and downstream neighbor(s) it may have in the QoS Segment. The *QosDevice* Service uses this knowledge along with its intrinsic knowledge of the L2 Technology—i.e., whether the L2 Technology requires the stream to be set up from the stream ingress or stream egress or both—to determine whether it is responsible for reserving resources on the QoS Segment or whether it should leave that responsibility to its upstream or downstream neighbor. If the *QosDevice* Service determines that it is not responsible for setting up the stream on the QoS Segment, it will return Successful completion with a *ReasonCode* = “001” and will register the TrafficDescriptor of the stream. The responsibility of QoS setup for each lower layer technology is found in [QD_Add]

A *QosManager* provides the lease time as part of a *TrafficDescriptor* as input to *AdmitTrafficQos()*. The *QosManager* received this lease time in the *TrafficDescriptor* from the Control Point that invoked it. When the lease time expires, the *QosDevice* Service releases the QoS resources allocated to that *TrafficDescriptor*. The state of the *QosDevice* Service after a release triggered by lease time expiration is the same as if it had occurred as the result of a *ReleaseAdmittedQos()* action.

The *UpdateTrafficLeaseTime()* action provides the Control Point with a mechanism to update the lease time of a traffic stream.

A *QosDevice* Service optionally collects information about traffic streams (Rotameter) from its interfaces. The *QosDevice* Service can be queried for the information using the action *GetRotameterInformation()*. The collection of data may be controlled using the action *ConfigureRotameterObservation()*.

2.5.2 Prioritized QoS

Control Points designed for version 3 use QoS:3 actions. Control Points designed for version 2 still use QoS:2 actions (as they are not designed for the QoS:3 actions).

2.5.2.1 Prioritized QoS usage by Control Points designed for version 3

A *QosDevice* Service exposes its static QoS capabilities and run time QoS state through the *GetExtendedQosState()* action. The static capabilities include type of native QoS support, such as "Prioritized" or "BestEffort". Other capabilities include maximum PHY rate.

The *AdmitTrafficQos()* action provides a mechanism for the *QosManager* to request the device to provide network resources for a traffic stream identified by the *TrafficDescriptor*.

The *QosManager* can indicate to the *QosDevice* Service to release resources in two ways. The *QosManager* may either call the *ReleaseAdmittedQos()* action or simply not renew the lease time associated with the traffic stream.

A *QosManager* may provide a lease time as part of a *TrafficDescriptor* as input to *AdmitTrafficQos()*. When the lease time expires, the *QosDevice* Service releases the QoS resources allocated to that *TrafficDescriptor*. The state of the *QosDevice* Service is the same as if it were a *ReleaseAdmittedQos()* action. In case a *QosManager* does not specify the lease time, the QoS resources remain allocated until they are released by a *QosManager*.

2.5.2.2 Prioritized QoS usage by Control Points designed for version 2

A *QosDevice* Service exposes its static QoS capabilities through the *GetQosDeviceCapabilities()* action. The static capabilities include type of native QoS support, such as "Prioritized" or "BestEffort". Other capabilities include maximum PHY rate.

The run time QoS state of the device may be very different from what was advertised through the *GetQosDeviceCapabilities()* and this state is exposed through the *GetQosState()* action. The *GetQosState()* action provides information about the currently active traffic streams on the device using TrafficDescriptor structures.

The *SetupTrafficQos()* action provides a mechanism for the *QosManager* to request the device to provide network resources for a traffic stream identified by the *TrafficDescriptor*.

Race conditions may occur when different *QosManager*s use the actions *GetQosState()* and *SetupTrafficQos()*. To identify such conditions, the *QosDevice* Service provides a unique identification of its state named *QosStateId* in reply to the action *GetQosState()*. The *QosManager* provides *QosStateId* as input argument to *SetupTrafficQos()*. In case the *QosStateId* sent by the *QosManager* does not match the most recent *QosStateId* handed out by the device, the device responds to *SetupTrafficQos()* with error code 760. This mechanism is used only for QoS Version 2; the *AdmitTrafficQos()* action does not have the same issues with race conditions.

The *QosManager* can indicate to the *QosDevice* Service to release resources in two ways. The *QosManager* may either call the *ReleaseTrafficQos()* action or simply not renew the lease time associated with the traffic stream.

The *QosDevice* Service provides L2 reachability information to the *QosManager* through the *GetPathInformation* action. This information includes the link identifier, MAC address, bridging information and reachable MAC addresses for each link on the *QosDevice* Service. This is useful to help a *QosManager* determine the topology of the network and the path of the traffic streams.

A *QosDevice* Service optionally collects information about traffic streams (Rotameter) from its interfaces. The *QosDevice* Service can be queried for the information using the action *GetRotameterInformation()*. The collection of data may be controlled using the action *ConfigureRotameterObservation()*.

A *QosManager* may provide a lease time as part of a *TrafficDescriptor* as input to *SetupTrafficQos()*. When the lease time expires, the *QosDevice* Service releases the QoS resources allocated to that *TrafficDescriptor*. The state of the *QosDevice* Service is the same as if it were a *ReleaseTrafficQos()* action. In case a *QosManager* does not specify the lease time, the QoS resources remain allocated until they are released by a *QosManager*.

2.5.3 Hybrid QoS

A Control Point requests Hybrid QoS when it desires end-to-end Parameterized QoS but is willing to accept Prioritized QoS for QoS Segments on which Parameterized QoS is not available.

A UPnP *QosDevice* Service exposes its QoS capabilities and its current QoS state through the *GetExtendedQosState*() action. This action returns both its static properties (i.e., capabilities and states that are independent of the state of the device) and its dynamic

properties (i.e., capabilities and states that depend upon the state of the device at the given time the action is invoked). The information returned by this action provides the *QosManager* with a complete run time representation of the QoS-related operating characteristics and the state of the device. Based on this, the *QosManager* determines which QoS Segments support what type of QoS.

The *AdmitTrafficQos*() action provides a mechanism for the *QosManager* to request the *QosDevice* Service to reserve network resources for a traffic stream identified by the Traffic Descriptor. When making the admission request, the *QosManager* sets the *RequestedQosType* to Hybrid QoS. Based on the capabilities of the requested QoS Segment, the *QosDevice* Service performs appropriate steps to configure Parameterized QoS or Prioritized QoS.

3 XML Service Descriptions

```
<?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>GetPathInformation</name>
        <argumentList>
            <argument>
                <name>PathInformation</name>
                <direction>out</direction>
                <relatedStateVariable>PathInformation</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>GetRotameterInformation</name>
        <argumentList>
            <argument>
                <name>RequestedNumRotameterObservations</name>
                <direction>in</direction>

    <relatedStateVariable>A_ARG_TYPE_NumRotameterObservations</relatedStateVariable>
            </argument>
            <argument>
                <name>RotameterObservation</name>
                <direction>out</direction>
                <relatedStateVariable>A_ARG_TYPE_RotameterInformation</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>ConfigureRotameterObservation</name>
        <argumentList>
            <argument>
                <name>RequestedConfRotameterObservations</name>
                <direction>in</direction>

    <relatedStateVariable>A_ARG_TYPE_ConfRotameterObservations</relatedStateVariable>
            </argument>
            <argument>
                <name>MaxPossibleRotameterObservations</name>
                <direction>out</direction>

    <relatedStateVariable>A_ARG_TYPE_MaxPossibleRotameterObservations</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>GetQosDeviceCapabilities</name>
        <argumentList>
            <argument>
                <name>QosDeviceCapabilities</name>
                <direction>out</direction>
    <relatedStateVariable>A_ARG_TYPE_QosDeviceCapabilities</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
```

```
        <name>GetQosDeviceInfo</name>
        <argumentList>
            <argument>
                <name>TrafficDescriptor</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor</relatedStateVariable>
            </argument>
            <argument>
                <name>QosDeviceInfo</name>
                <direction>out</direction>
                <relatedStateVariable>A_ARG_TYPE_QosDeviceInfo</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>GetQosState</name>
        <argumentList>
            <argument>
                <name>QosDeviceState</name>
                <direction>out</direction>
                <relatedStateVariable>A_ARG_TYPE_QosDeviceState</relatedStateVariable>
            </argument>
            <argument>
                <name>NumberOfTrafficDescriptors</name>
                <direction>out</direction>
                <relatedStateVariable>A_ARG_TYPE_NumTrafficDescriptors</relatedStateVariable>
            </argument>
            <argument>
                <name>ListOfTrafficDescriptors</name>
                <direction>out</direction>

    <relatedStateVariable>A_ARG_TYPE_TrafficDescriptorsPerInterface</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>ReleaseTrafficQos</name>
        <argumentList>
            <argument>
                <name>ReleaseTrafficHandle</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_TrafficHandle</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>SetupTrafficQos</name>
        <argumentList>
            <argument>
                <name>SetupTrafficDescriptor</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor</relatedStateVariable>
            </argument>
            <argument>
                <name>QosStateId</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_QosStateId</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>GetExtendedQosState</name>
        <argumentList>
            <argument>
                <name>InputTrafficDescriptor</name>
                <direction>in</direction>

    <relatedStateVariable>A_ARG_TYPE_TrafficDescriptorContainer</relatedStateVariable>
            </argument>
            <argument>
                <name>TrafficDescriptorsWanted</name>
                <direction>in</direction>

    <relatedStateVariable>A_ARG_TYPE_TrafficDescriptorsWanted</relatedStateVariable>
            </argument>
            <argument>
                <name>QosDeviceExtendedState</name>
                <direction>out</direction>
                <relatedStateVariable>A_ARG_TYPE_QosDeviceExtendedState</relatedStateVariable>
            </argument>
            <argument>
                <name>QosDeviceInfo</name>
                <direction>out</direction>
```

```
            <relatedStateVariable>A_ARG_TYPE_QosDeviceInfoContainer</relatedStateVariable>
        </argument>
        <argument>
            <name>ListOfAdmittedTraffic</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_ListOfAdmittedTraffic</relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>SetPreferredQph</name>
    <argumentList>
        <argument>
            <name>PreferredQph</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_PreferredQph</relatedStateVariable>
        </argument>
        <argument>
            <name>CurrentPreferredQph</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_PreferredQph</relatedStateVariable>
        </argument>
        <argument>
            <name>SetPreferredQphResults</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_SetPreferredQphResults</relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>AdmitTrafficQos</name>
    <argumentList>
        <argument>
            <name>AdmitTrafficDescriptor</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor</relatedStateVariable>
        </argument>
        <argument>
            <name>Resource</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_Resource</relatedStateVariable>
        </argument>
        <argument>
            <name>AdmitTrafficQosSucceeded</name>
            <direction>out</direction>

<relatedStateVariable>A_ARG_TYPE_AdmitTrafficQosSucceeded</relatedStateVariable>
        </argument>
        <argument>
            <name>AdmitTrafficQosExtendedResult</name>
            <direction>out</direction>

<relatedStateVariable>A_ARG_TYPE_AdmitTrafficQosExtendedResult</relatedStateVariable>
        </argument>
        <argument>
            <name>Layer2MappingContainer</name>
            <direction>out</direction>
            <relatedStateVariable>A_ARG_TYPE_Layer2MappingContainer</relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>UpdateAdmittedQos</name>
    <argumentList>
        <argument>
            <name>UpdateTrafficDescriptor</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficDescriptor</relatedStateVariable>
        </argument>
        <argument>
            <name>Resource</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_Resource</relatedStateVariable>
        </argument>
        <argument>
            <name>PreemptingTrafficInfo</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_PreemptingTrafficInfo</relatedStateVariable>
        </argument>
        <argument>
            <name>AdmitTrafficQosSucceeded</name>
            <direction>out</direction>
```

```
    <relatedStateVariable>A_ARG_TYPE_AdmitTrafficQosSucceeded</relatedStateVariable>
            </argument>
            <argument>
                <name>AdmitTrafficQosExtendedResult</name>
                <direction>out</direction>

    <relatedStateVariable>A_ARG_TYPE_AdmitTrafficQosExtendedResult</relatedStateVariable>
            </argument>
            <argument>
                <name>Layer2Mapping</name>
                <direction>out</direction>
                <relatedStateVariable>A_ARG_TYPE_Layer2MappingContainer</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>ReleaseAdmittedQos</name>
        <argumentList>
            <argument>
                <name>ReleaseTrafficHandle</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_TrafficHandle</relatedStateVariable>
            </argument>
            <argument>
                <name>Resource</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_Resource</relatedStateVariable>
            </argument>
            <argument>
                <name>PreemptingTrafficInfo</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_PreemptingTrafficInfo</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>GetUnexpectedStreamChanges</name>
        <argumentList>
            <argument>
                <name>NumberOfUnexpectedStreamChangesRequested</name>
                <direction>in</direction>

    <relatedStateVariable>A_ARG_TYPE_NumberOfUnexpectedStreamChangesRequested</relatedStateVar
iable>
            </argument>
            <argument>
                <name>NumberOfUnexpectedStreamChangesReported</name>
                <direction>out</direction>

    <relatedStateVariable>A_ARG_TYPE_NumberOfUnexpectedStreamChangesReported</relatedStateVari
able>
            </argument>
            <argument>
                <name>MostRecentUnexpectedStreamChanges</name>
                <direction>out</direction>

    <relatedStateVariable>A_ARG_TYPE_ListOfMostRecentUnexpectedStreamChanges</relatedStateVari
able>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>VerifyTrafficHandle</name>
        <argumentList>
            <argument>
                <name>TrafficHandleToVerify</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_TrafficHandle</relatedStateVariable>
            </argument>
            <argument>
                <name>TrafficDescriptorWanted</name>
                <direction>in</direction>

    <relatedStateVariable>A_ARG_TYPE_TrafficDescriptorsWanted</relatedStateVariable>
            </argument>
            <argument>
                <name>TrafficDescriptor</name>
                <direction>out</direction>

    <relatedStateVariable>A_ARG_TYPE_TrafficDescriptorContainer</relatedStateVariable>
            </argument>
        </argumentList>
```

```
</action>
<action>
    <name>UpdateTrafficLeaseTime</name>
    <argumentList>
        <argument>
            <name>TrafficHandle</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_TrafficHandle</relatedStateVariable>
        </argument>
        <argument>
            <name>NewTrafficLeaseTime</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_NewTrafficLeaseTime</relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>SetL2Map</name>
    <argumentList>
        <argument>
            <name>Layer2MappingContainer</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_Layer2Mapping</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_TrafficHandle</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_QosStateId</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_QosDeviceState</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_NumTrafficDescriptors</name>
        <dataType>ui4</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_QosDeviceInfo</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
        <name>PathInformation</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_QosDeviceCapabilities</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_TrafficDescriptorsPerInterface</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_NumRotameterObservations</name>
        <dataType>ui4</dataType>
        <defaultValue>1</defaultValue>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_RotameterInformation</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_ConfRotameterObservations</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
        <name>MostRecentStreamAction</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_MaxPossibleRotameterObservations</name>
        <dataType>ui4</dataType>
```

```
            <defaultValue>1</defaultValue>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_TrafficDescriptorsWanted</name>
            <dataType>boolean</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_QosDeviceExtendedState</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_ListOfAdmittedTraffic</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_PreferredQph</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_SetPreferredQphResults</name>
            <dataType>ui4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_Resource</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_AdmitTrafficQosExtendedResult</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="yes">
            <name>UnexpectedStreamChange</name>
            <dataType>ui4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_PreemptingTrafficInfo</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_ListOfMostRecentUnexpectedStreamChanges</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_Layer2Mapping</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_AdmitTrafficQosSucceeded</name>
            <dataType>boolean</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_NumberOfUnexpectedStreamChangesRequested</name>
            <dataType>ui4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_NumberOfUnexpectedStreamChangesReported</name>
            <dataType>ui4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_NewTrafficLeaseTime</name>
            <dataType>ui4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_TrafficDescriptorContainer</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_Layer2MappingContainer</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_QosDeviceInfoContainer</name>
            <dataType>string</dataType>
        </stateVariable>
        Declarations for other state variables added by UPnP vendor (if any) go here
    </serviceStateTable>
</scpd>
</serviceStateTable>
</scpd>
```

4 Test

No semantic tests have been specified for this service.

Annex A
(informative)
Additional Examples for State Variables

This contains additional examples for some of the State Variables.

A.1 Additional *PathInformation* Examples

A.1.1 Sample argument XML string – PC with two network interfaces that are both end point device and bridged

This is an example of an end point network device with two network interfaces. This device forwards L2 frames between the two network interfaces.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceReachableMacs
  xmlns="http://www.upnp.org/schemas/PathInformation.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/PathInformation.xsd
http://www.upnp.org/schemas/qos/PathInformation-v2.xsd">
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <MacAddress>112233aabb03</MacAddress>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth1</LinkId>
      <MacAddress>112233aabb02</MacAddress>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth2</LinkId>
      <MacAddress>112233aabb32</MacAddress>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb03</ReachableMac>
      <ReachableMac>112233aabb02</ReachableMac>
      <ReachableMac>112233aabb01</ReachableMac>
      <ReachableMac>112233aabb04</ReachableMac>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth3</LinkId>
      <MacAddress>112233aabb14</MacAddress>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb05</ReachableMac>
  </LinkReachableMacs>
</DeviceReachableMacs>
```

A.1.2 Sample argument XML string –Four port Ethernet Switch

This is an example of a L2 switching device that interconnects four physical Ethernet ports. The device supports L2 frame forwarding between all ports.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceReachableMacs
  xmlns="http://www.upnp.org/schemas/PathInformation.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/PathInformation.xsd
http://www.upnp.org/schemas/qos/PathInformation-v2.xsd">
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb03</ReachableMac>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth1</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb07</ReachableMac>
      <ReachableMac>112233aabb05</ReachableMac>
  </LinkReachableMacs>
  <LinkReachableMacs>
```

```
      <LinkId>eth2</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb02</ReachableMac>
      <ReachableMac>112233aabb01</ReachableMac>
      <ReachableMac>112233aabb04</ReachableMac>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth3</LinkId>
      <BridgeId>Bridge0</BridgeId>
  </LinkReachableMacs>
</DeviceReachableMacs>
```

A.1.3 Sample argument XML string – Wireless AP with one Ethernet Interface

This is an example of a wireless access point with three associated wireless stations and a single Ethernet port. The device supports L2 frame forwarding between all links. This includes forwarding between wireless stations or to the Ethernet interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceReachableMacs
  xmlns="http://www.upnp.org/schemas/PathInformation.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/PathInformation.xsd
http://www.upnp.org/schemas/qos/PathInformation-v2.xsd">
  <LinkReachableMacs>
      <LinkId>WL0</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb02</ReachableMac>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>WL1</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb01</ReachableMac>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>WL2</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb04</ReachableMac>
      <ReachableMac>112233aabb09</ReachableMac>
  </LinkReachableMacs>
  <LinkReachableMacs>
     <LinkId>eth0</LinkId>
     <BridgeId>Bridge0</BridgeId>
     <ReachableMac>112233aabb03</ReachableMac>
     <ReachableMac>112233aabb07</ReachableMac>
     <ReachableMac>112233aabb05</ReachableMac>
  </LinkReachableMacs>
</DeviceReachableMacs>
```

A.1.4 Sample argument XML string – Bridge device between Wireless station and Ethernet

This is an example of a bridging device with two interfaces on different L2 Technologies. It does L2 forwarding of frames between wireless station interface and the wired Ethernet interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceReachableMacs
  xmlns="http://www.upnp.org/schemas/PathInformation.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/PathInformation.xsd
http://www.upnp.org/schemas/qos/PathInformation-v2.xsd">
  <LinkReachableMacs>
      <LinkId>WL0</LinkId>
      <BridgeId>Bridge0</BridgeId>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb04</ReachableMac>
  </LinkReachableMacs>
```

```
</DeviceReachableMacs>
```

A.2 Additional A_ARG_TYPE_RotameterInformation Examples

A.2.1 Sample argument XML string – PC with two network interfaces that are both end point devices

Similar to the previous example this is an example of an end point network device with two network interfaces. In this example the interfaces are actively connected and actively making Rotameter Observations. The example also shows per stream rotameter measurements over the wireless segment.

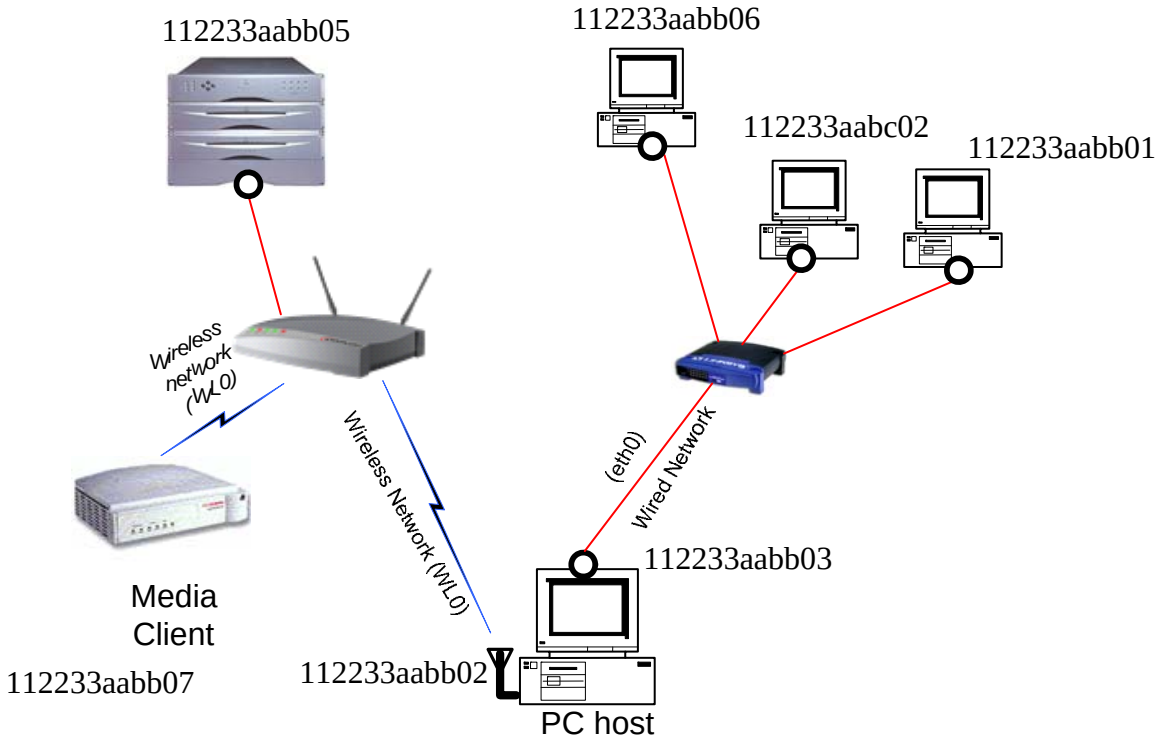


Figure A.1 — Example of a PC connected to an active network

```
<?xml version="1.0" encoding="UTF-8"?>
<RotameterInformation
xsi:schemaLocation="http://www.upnp.org/schemas/RotameterInformation.xsd
RotameterInformation-v3.xsd"
xmlns="http://www.upnp.org/schemas/RotameterInformation.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <MacAddress>112233aabb03</MacAddress>
      <RotameterObservation>
         <RotameterIndex>10000001</RotameterIndex>
         <ROAddr>112233aabb03</ROAddr>
         <ROBits>1000000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:03:23-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10000002</RotameterIndex>
         <ROAddr>112233aabb03</ROAddr>
         <ROBits>1000000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:23-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
```

```
    <RotameterObservation>
       <RotameterIndex>10000006</RotameterIndex>
       <ROAddr>112233aabb06</ROAddr>
       <ROBits>500000</ROBits>
       <ROPeriod>1</ROPeriod>
       <ReportingDateTime>2004-11-26T15:03:23-08:00</ReportingDateTime>
       <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
    </RotameterObservation>
    <RotameterObservation>
       <RotameterIndex>10000007</RotameterIndex>
       <ROAddr>112233aabb06</ROAddr>
       <ROBits>500000</ROBits>
       <ROPeriod>1</ROPeriod>
       <ReportingDateTime>2004-11-26T15:04:23-08:00</ReportingDateTime>
       <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
    </RotameterObservation>
    <RotameterObservation>
       <RotameterIndex>10000011</RotameterIndex>
       <ROAddr>112233aabc02</ROAddr>
       <ROBits>500000</ROBits>
       <ROPeriod>1</ROPeriod>
       <ReportingDateTime>2004-11-26T15:03:43-08:00</ReportingDateTime>
       <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
    </RotameterObservation>
    <RotameterObservation>
       <RotameterIndex>10000012</RotameterIndex>
       <ROAddr>112233aabc02</ROAddr>
       <ROBits>500000</ROBits>
       <ROPeriod>1</ROPeriod>
       <ReportingDateTime>2004-11-26T15:04:43-08:00</ReportingDateTime>
       <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
    </RotameterObservation>
    <RotameterObservation>
       <RotameterIndex>10000013</RotameterIndex>
       <ROAddr>112233aabc02</ROAddr>
       <ROBits>500000</ROBits>
       <ROPeriod>1</ROPeriod>
       <ReportingDateTime>2004-11-26T15:05:43-08:00</ReportingDateTime>
       <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
    </RotameterObservation>
</LinkReachableMacs>
<LinkReachableMacs>
    <LinkId>WL0</LinkId>
    <BridgeId>String</BridgeId>
    <v3>
       <QosSegmentId>274A9172877A91C2DE</QosSegmentId>
       <MacAddress>112233aabb07</MacAddress>
    </v3>
    <RotameterObservation>
       <RotameterIndex>10000021</RotameterIndex>
       <ROAddr>112233aabb05</ROAddr>
       <ROBits>1000000</ROBits>
       <ROPeriod>1</ROPeriod>
       <ReportingDateTime>2004-11-26T15:03:23-08:00</ReportingDateTime>
       <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
       <v3>
          <ROBitsParameterized>500000</ROBitsParameterized>
          <ROPacketsParameterized>3500</ROPacketsParameterized>
          <ROParameterizedPacketsDropped>20</ROParameterizedPacketsDropped>
          <ROPerStreamInformation>
             <TrafficHandle>TH-882771240-777273</TrafficHandle>
             <Layer2StreamId>
                0000000000000000000000000000000000000000000000000000000000482d2a1
             </Layer2StreamId>
             <StreamBitsTransmitted>250000</StreamBitsTransmitted>
             <StreamPacketsTransmitted>1750</StreamPacketsTransmitted>
             <StreamBitsReceived>200000</StreamBitsReceived>
             <StreamPacketsReceived>1000</StreamPacketsReceived>
             <StreamPacketsDropped>10</StreamPacketsDropped>
          </ROPerStreamInformation>
          <ROPerStreamInformation>
             <TrafficHandle>TH-882771240-777273</TrafficHandle>
             <Layer2StreamId>
                0000000000000000000000000000000000000000000000000000000000582d2a1
```

```
            </Layer2StreamId>
            <StreamBitsTransmitted>250000</StreamBitsTransmitted>
            <StreamPacketsTransmitted>1750</StreamPacketsTransmitted>
            <StreamBitsReceived>200000</StreamBitsReceived>
            <StreamPacketsReceived>1000</StreamPacketsReceived>
            <StreamPacketsDropped>10</StreamPacketsDropped>
         </ROPerStreamInformation>
      </v3>
   </RotameterObservation>
   <RotameterObservation>
      <RotameterIndex>10000022</RotameterIndex>
      <ROAddr>112233aabb05</ROAddr>
      <ROBits>1000000</ROBits>
      <ROPeriod>1</ROPeriod>
      <ReportingDateTime>2004-11-26T15:04:23-08:00</ReportingDateTime>
      <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      <v3>
         <ROBitsParameterized>400000</ROBitsParameterized>
         <ROPacketsParameterized>2500</ROPacketsParameterized>
         <ROParameterizedPacketsDropped>30</ROParameterizedPacketsDropped>
         <ROPerStreamInformation>
            <TrafficHandle>TH-992771240-777273</TrafficHandle>
            <Layer2StreamId>
               0000000000000000000000000000000000000000000000000000001482d2a1
            </Layer2StreamId>
            <StreamBitsTransmitted>200000</StreamBitsTransmitted>
            <StreamPacketsTransmitted>1000</StreamPacketsTransmitted>
            <StreamBitsReceived>150000</StreamBitsReceived>
            <StreamPacketsReceived>1000</StreamPacketsReceived>
            <StreamPacketsDropped>40</StreamPacketsDropped>
         </ROPerStreamInformation>
         <ROPerStreamInformation>
            <TrafficHandle>TH-772771240-777273</TrafficHandle>
            <Layer2StreamId>
               0000000000000000000000000000000000000000000000000000026482d2a1
            </Layer2StreamId>
            <StreamBitsTransmitted>200000</StreamBitsTransmitted>
            <StreamPacketsTransmitted>1000</StreamPacketsTransmitted>
            <StreamBitsReceived>150000</StreamBitsReceived>
            <StreamPacketsReceived>1000</StreamPacketsReceived>
            <StreamPacketsDropped>30</StreamPacketsDropped>
         </ROPerStreamInformation>
      </v3>
   </RotameterObservation>
   <RotameterObservation>
      <RotameterIndex>10000013</RotameterIndex>
      <ROAddr>112233aabb05</ROAddr>
      <ROBits>380000</ROBits>
      <ROPeriod>1</ROPeriod>
      <ReportingDateTime>2004-11-26T15:03:24-08:00</ReportingDateTime>
      <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
   </RotameterObservation>
   <RotameterObservation>
      <RotameterIndex>10000031</RotameterIndex>
      <ROAddr>112233aabb07</ROAddr>
      <ROBits>500000</ROBits>
      <ROPeriod>1</ROPeriod>
      <ReportingDateTime>2004-11-26T15:03:43-08:00</ReportingDateTime>
      <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
   </RotameterObservation>
   <RotameterObservation>
      <RotameterIndex>10000032</RotameterIndex>
      <ROAddr>112233aabb07</ROAddr>
      <ROBits>500000</ROBits>
      <ROPeriod>1</ROPeriod>
      <ReportingDateTime>2004-11-26T15:04:43-08:00</ReportingDateTime>
      <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
   </RotameterObservation>
   <RotameterObservation>
      <RotameterIndex>10000033</RotameterIndex>
      <ROAddr>112233aabb07</ROAddr>
      <ROBits>500000</ROBits>
      <ROPeriod>1</ROPeriod>
      <ReportingDateTime>2004-11-26T15:05:43-08:00</ReportingDateTime>
```

```
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
   </LinkReachableMacs>
</RotameterInformation>
```

A.2.2 Sample argument XML string – PC with two network interfaces that are both end point device with TrafficImportanceNumber reporting

Similar to the previous example this is an example of an end point network device with two actively connected network interfaces. In this example, one interface reports bits per interpreted TrafficImportanceNumber.

```
<?xml version="1.0" encoding="UTF-8"?>
<RotameterInformation
  xmlns="http://www.upnp.org/schemas/RotameterInformation.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/RotameterInformation.xsd
http://www.upnp.org/schemas/qos/RotameterInformation-v2.xsd">
   <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <MacAddress>112233aabb03</MacAddress>
      <RotameterObservation>
         <RotameterIndex>10000001</RotameterIndex>
         <ROAddr>112233aabb03</ROAddr>
         <ROBits>1000000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:03:23-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10000002</RotameterIndex>
         <ROAddr>112233aabb03</ROAddr>
         <ROBits>1000000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:23-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10000006</RotameterIndex>
         <ROAddr>112233aabb06</ROAddr>
         <ROBits>500000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:03:23-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10000007</RotameterIndex>
         <ROAddr>112233aabb06</ROAddr>
         <ROBits>500000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:23-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10000011</RotameterIndex>
         <ROAddr>112233aabc02</ROAddr>
         <ROBits>500000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:03:43-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10000012</RotameterIndex>
         <ROAddr>112233aabc02</ROAddr>
         <ROBits>500000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:43-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
  </LinkReachableMacs>
</RotameterInformation>
```

A.2.3 Sample argument XML string –Four port Ethernet Switch

This is an example of a L2 switching device that interconnects four physical Ethernet ports. The device supports L2 frame forwarding between all ports.

```
<?xml version="1.0" encoding="UTF-8"?>
<RotameterInformation
  xmlns="http://www.upnp.org/schemas/RotameterInformation.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/RotameterInformation.xsd
http://www.upnp.org/schemas/qos/RotameterInformation-v2.xsd">
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <RotameterObservation>
         <RotameterIndex>10000001</RotameterIndex>
         <ROAddr>112233aabb03</ROAddr>
         <ROBits>1000000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:03:23-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth1</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <RotameterObservation>
         <RotameterIndex>10000004</RotameterIndex>
         <ROAddr>112233aabb06</ROAddr>
         <ROBits>1000000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:23-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10000007</RotameterIndex>
         <ROAddr>112233aabb01</ROAddr>
         <ROBits>1000000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:43-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth2</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <RotameterObservation>
         <RotameterIndex>10000017</RotameterIndex>
         <ROAddr>112233aabc02</ROAddr>
         <ROBits>2300000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:33-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth3</LinkId>
      <BridgeId>Bridge0</BridgeId>
  </LinkReachableMacs>
</RotameterInformation>
```

A.2.4 Sample argument XML string – Wireless AP with one Ethernet Interface

This is an example of a wireless access point with three associated wireless stations and a single Ethernet port. The device supports L2 frame forwarding between all links. This includes forwarding (bridging) between wireless stations and to the Ethernet interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<RotameterInformation
  xmlns="http://www.upnp.org/schemas/RotameterInformation.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/RotameterInformation.xsd
```

```
http://www.upnp.org/schemas/qos/RotameterInformation-v2.xsd">
  <LinkReachableMacs>
      <LinkId>WL0</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <RotameterObservation>
         <RotameterIndex>10001001</RotameterIndex>
         <ROAddr>112233aabb02</ROAddr>
         <ROBits>2000000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:43-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10001002</RotameterIndex>
         <ROAddr>112233aabb07</ROAddr>
         <ROBits>2300000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:33-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <RotameterObservation>
         <RotameterIndex>10001004</RotameterIndex>
         <ROAddr>112233aabb05</ROAddr>
         <ROBits>5800000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:03:34-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10001005</RotameterIndex>
         <ROAddr>112233aabb07</ROAddr>
         <ROBits>3700000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:34-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
      <RotameterObservation>
         <RotameterIndex>10001006</RotameterIndex>
         <ROAddr>112233aabb05</ROAddr>
         <ROBits>6200000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:31-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
  </LinkReachableMacs>
</RotameterInformation>
```

A.2.5 Sample argument XML string – Bridge device between Wireless station and Ethernet

This is an example of a bridging device with two interfaces on different L2 Technologies. It does L2 forwarding of frames between wireless station interface and the wired Ethernet interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<RotameterInformation
  xmlns="http://www.upnp.org/schemas/RotameterInformation.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.upnp.org/schemas/RotameterInformation.xsd
http://www.upnp.org/schemas/qos/RotameterInformation-v2.xsd">
  <LinkReachableMacs>
      <LinkId>WL0</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <RotameterObservation>
         <RotameterIndex>10001004</RotameterIndex>
         <ROAddr>112233aabb02</ROAddr>
         <ROBits>5800000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:03:34-08:00</ReportingDateTime>
```

```
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <RotameterObservation>
         <RotameterIndex>10001005</RotameterIndex>
         <ROAddr>112233aabb03</ROAddr>
         <ROBits>3700000</ROBits>
         <ROPeriod>1</ROPeriod>
         <ReportingDateTime>2004-11-26T15:04:34-08:00</ReportingDateTime>
         <MonitorResolutionPeriod>60</MonitorResolutionPeriod>
      </RotameterObservation>
    </LinkReachableMacs>
</RotameterInformation>
```

Annex B
(normative)
Template for Requirements on the QosDevice Service implementation that are specific for the underlying Network Technologies

UPnP-QoS relies on Layer 2 Technologies for much of its QoS functionality. UPnP-QoS is designed to work with existing Layer 2 Technologies and is flexible enough to accomodate both future versions of existing technologies and future technologies. For each Layer 2 Technology, an addendum must be provided to explain, at a minimum, the mapping of the various UPnP-QoS functions and parameters to the Layer 2 Technology. This ensures that all implementations of a particular Layer 2 Technology provide the mappings in the same way. These Layer 2 Technology addendums are collected together in an addendum to the QosDevice Service document [QD_Add] .

To supply an addendum, an author MUST start with this template. Each clause of the template must be filled in to ensure valid operation of the UPnP-QoS system on that Layer 2 Technology. Instructions are present in each clause to assist the author in completing it. <all text in this format are instructions to the addendum writer and should be removed from the final document>

An addendum MUST be provided for each Layer 2 Technology that supports UPnP-QoS. In addition, an addendum may be provided for a "heterogeneous" Layer 2 Technology QoS Segment, one that encompasses two or more different Layer 2 Technologies. The heterogeneous QoS Segment is a single QoS Segment, where QoS Devices that fall under a heterogeneous QoS Segment MUST have a common way to define the QosSegmentId, and are, by definition QoS Devices on the same QoS segment. A heterogeneous QoS Segment is managed by the QoS Manager as a single QoS Segment.

B.1 <Technology Name>

B.1.1 References

[abbrev] Full bibliograhical data of the main specification(s) for the technology.

B.1.2 Priority Mapping

<Paragraph explaining how priorities in the underlying technology are mapped to UPnP-QoS Traffic Importance Numbers>

Table B.1 — Priority Mapping

QoS Traffic Importance Number	*<technology>*
0	
1	
2	
3	
4	
5	
6	
7	

B.1.3 *QosSegmentId* formation

<For technologies with a registered IANA Technology Type:>

The *QosSegmentId* MUST be formed by concatenating the IANATechnologyType "*IANATechnologyType*" (fill in the actual number of the *IANATechnologyType* or fill in "NaN" if the technology does not have an *IANATechnoloyType*) with *some value that is unique to each individual L2 segment. <If the appropriate IANATechnologyType is already in use by another technology in this addendum, use that IANATechnologyType with a single letter (that is also not in use by another technology in this addendum) appended to it to guarantee uniqueness.> <When using Hexdigits:> Where each hex digit MUST correspond to a specific nibble with the following restrictions: The least significant nibble MUST be the leftmost nibble in the QosSegmentId following the IANAifType. Within the nibble, the least significant bit MUST correspond to the least significant bit as defined in XML/UPnP.*

Example: *zzzzz*

<For technologies that do not have a registered IANA Technology Type:>

The *QosSegmentId* MUST be formed by concatenating first the string "NaN", second a domain name that is associated with the technology, third a colon (":") and fourth some unique value that is unique to devices in that segment.

Example: "NaN:example.org:*zzzz*"

<All devices in the same QosSegment SHALL define the QosSegmentId in the same way so that a string compare returns equivalence. QosSegmentIds of 2 different segments in the same LAN MUST return DIFFERENT QosSegmentIds to ensure QosManagers and/or other upper layers can differentiate different segments in a path..>

B.1.4 *Layer2StreamId* representation

The *Layer2StreamId* MUST be a string of *64 characters* . (Make sure this is unique on this QoS Segment)

Example: "*0000000000000001000000000000000100000000000000010000000000000001*"

B.1.5 Mapping of UPnP-QoS Parameters to *<technology>* Parameters

Table B.2 shows how *<technology name>* Traffic specification parameters are determined from UPnP-QoS TSPEC parameters.

<note: The table now contains all UPnP-QoS parameters. The table should be edited such that in the left column the technology's parameter names are listed and in the third column how this technology's parameter's are determined from one (or more) UPnP-QoS parameters. This is not necessarily a two-way one-to-one mapping. Note that TrafficLeaseTime is not an element of the TrafficSpecification. Elements in the <technology> parameter are expressed in the units defined in the underlying technology. The calculations in the "As calculated from UPnP-QoS Parameter(s)" column must indicate the proper unit conversion from UPnP-QoS units.

Table B.2 — Traffic Specification Parameters

<technology> Parameter	R/O	As calculated from UPnP-QoS parameter(s)	Comment
		RequestedQosType	
		TrafficClass	
		DataRate	
		MinServiceRate	
		MaxBurstSize	
		PeakDataRate	
		ReservedServiceRate	
		TimeUnit	
		MaxPacketSize	
		E2EMaxDelayHigh	
		E2EMaxDelayLow	
		E2EMaxJitter	
		QosSegmentMaxDelayHigh	
		QosSegmentMaxDelayLow	
		QosSegmentMaxJitter	
		MaxServiceInterval	
		MinServiceInterval	
		LossSensitivity	
		ServiceType	
		TrafficLeaseTime	

<The R/O column MUST be set to "R" for any parameter that is required for successful admission by the <technology name> admission procedure>.

B.1.6 Blocking traffic stream identification

(If the technology supports indentification of blocking streams, then include the following, otherwise remove clause):

If an *AdmitTrafficQos()* or *UpdateAdmittedQos()* action fails to admit a traffic stream because of inadequate resources on a QoS Segment, the *QosDevice* Service MUST return a list containing the *Layer2StreamId* values of all currently active connections *<add more information how to obtain this list>.*

B.1.7 Responsibility for QoS Setup

<Select one of the following 5 that is appropriate. The exact wording in the following paragraphs may not be appropriate for some technologies. Text may be edited as necessary.>

In <technology> the source within the QoS Segment is responsible for the setup of QoS. Thus, if a *QosDevice* Service receives an *AdmitTrafficQos()* action or an *UpdateAdmittedQos()* action in which the *Resource* argument indicates that there is a *QosDevice* Service upstream from this *QosDevice* Service, it MUST NOT take any action but MUST acknowledge the request by returning a *ReasonCode* = "001" to the QoS Manager.

In <technology> the destination within the QoS Segment is responsible for the setup of QoS. Thus, if a *QosDevice* Service receives an *AdmitTrafficQos()* action or an *UpdateAdmittedQos()* action in which the *Resource* argument indicates that there is a

QosDevice Service downstream from this *QosDevice* Service, it MUST NOT take any action but MUST acknowledge the request by returning a *ReasonCode* = “001” to the QoS Manager.

In <technology> the source or the destination within the QoS Segment is responsible for the setup of QoS and for UPnP-QoS setup the arbitrary choice was made to set up QoS from the destination side. Thus, if a *QosDevice* Service receives an *AdmitTrafficQos()* action or an *UpdateAdmittedQos()* action in which the *Resource* argument indicates that there is a *QosDevice* Service downstream from this *QosDevice* Service, it MUST NOT take any action but MUST acknowledge the request by returning a *ReasonCode* = “001” to the QoS Manager.

In <technology> the source or the destination within the QoS Segment is responsible for the setup of QoS and for UPnP-QoS setup the arbitrary choice was made to set up QoS from the source side. Thus, if a *QosDevice* Service receives an *AdmitTrafficQos()* action or an *UpdateAdmittedQos()* action in which the *Resource* argument indicates that there is a *QosDevice* Service upstream from this *QosDevice* Service, it MUST NOT take any action but MUST acknowledge the request by returning a *ReasonCode* = “001” to the QoS Manager.

In <technology> the source and destination point of the QoS Segment are jointly responsible for the setup of QoS. Thus, if a *QosDevice* Service receives an *AdmitTrafficQos()* action or an *UpdateAdmittedQos()* in which the *Resource* argument indicates that there is a *QosDevice* Service both upstream and downstream from this *QosDevice* Service, it MUST NOT take any action but MUST acknowledge the request by returning a *ReasonCode* = “001” to the QoS Manager.

B.1.7.1 SetL2Map Requirements

<Delete this clause if there are no requirements>

A *QosDevice* Service with a *<technology name>* interface MUST implement the *QD:SetL2Map()* action.

<Select at most one of the two following sentences. The requirement needs to be conistent with the prior choice of which end sets up QoS>

A *QosDevice* Service with a *<technology name>* interface that is the source of the traffic stream in the *<technology name>* QoS segment MUST provide a valid *Layer2StreamId* in the *QD:AdmitTrafficQos()* and *QD:UpdateAdmittedQos()* actions. (The QosManager MUST subsequently provide this *Layer2StreamId* to the sink of the traffic stream in this QoS segment.)

A *QosDevice* Service with a *<technology name>* interface that is the sink of the traffic stream in the *<technology name>* QoS segment MUST provide a valid *Layer2StreamId* in the *QD:AdmitTrafficQos()* and *QD:UpdateAdmittedQos()* actions. (The QosManager MUST subsequently provide this *Layer2StreamId* to the source of the traffic stream in this QoS segment.)

B.1.8 Mapping of *<technology>* Returned Parameters to *ProtoTspec* Parameters

Table B.3 shows how UPnP-QoS ProtoTspec parameters are determined from returned *<technology name>* parameters. (See *QosManager* Service for definition of ProtoTspec)

<note: The table now contains all UPnP-QoS parameters. The table should be edited to include only parameters returned by the indicated technology. The middle column should indicate how this UPnP-QoS parameter value is determined from the technology’s parameters. This table should only be used for parameters that are determined in a common way on all implementations for a particular technology. Parameters whose values are implementation or vendor specific should not appear in this table. Elements in the UPnP-QoS parameter are expressed in the units defined in the ProtoTspec definition. The

calculations in the "As calculated from <technology> Parameter(s)" column must indicate the proper unit conversion.

Table B.3 — ProtoTspec Parameters

UPnP-QoS parameter	*As calculated from <technology>* Parameter(s)	Comment
RequestedQosType		
TrafficClass		
DataRate		
MinServiceRate		
MaxBurstSize		
PeakDataRate		
ReservedServiceRate		
TimeUnit		
MaxPacketSize		
QosSegmentMaxDelayHigh		
QosSegmentMaxDelayLow		
QosSegmentMaxJitter		
MaxServiceInterval		
MinServiceInterval		
LossSensitivity		
ServiceType		

B.1.9 Mapping of *<technology>* Returned Parameters to *AdmitTrafficQosExtendedResult and AllocatedResources* Parameters

Table B.4 shows how UPnP-QoS *AdmitTrafficQosExtendedResult* and *AllocatedResources* parameters are determined from returned *<technology name>* parameters.

<Note: The template table contains all parameters for the two structures. The table should be edited to include only parameters returned by the indicated technology. The middle column should indicate how this UPnP-QoS parameter value is determined from the technology's parameters.This table should only be used for parameters that are determined in a common way on all implementations for a particular technology. Parameters whose values are implementation or vendor specific should not appear in this table. Elements in the UPnP-QoS parameter are expressed in the units defined in the AdmitTrafficQosExtendedResult and AllocatedResources definitions. The calculations in the "As calculated from <technology> Parameter(s)" column must indicate the proper unit conversion.

Technologies that return values for MaxCommittedDelay and MaxCommittedJitter, return the delay or jitter for the interface on an individual device or for a segment Each technology selects one of three ways to return these values. By each individual QosDevice, by the source of a QoS Segment or by the sink of a QoS Segment. Additional delay and jitter from processing above the interface(s) is added by the QosDevice Service before returning the value to the QosManager. See clause 2.4.9.2.1 for a complete discussion of delay and jitter.

Suggested Text for delay (pick one or modify as necessary):

A *QosDevice* Service with a *<technology name>* interface MUST return only its contribution to the *MaxCommittedDelay* value for its QoS Segment. All other QosDevices on the QoS Segment MUST also return their own contribution.

A *QosDevice* Service with a *<technology name>* interface that is the source of the traffic stream in the *<technology name>* QoS segment MUST return the sum of *MaxCommittedDelay* values for all the QosDevices on the QoS Segment. All other QosDevices on the QoS Segment MUST return zero or any delay value not reported by the source device.

A *QosDevice* Service with a *<technology name>* interface that is the sink of the traffic stream in the *<technology name>* QoS segment MUST return the sum of *MaxCommittedDelay* values for all the QosDevices on the QoS Segment. All other QosDevices on the QoS Segment MUST return zero or any delay value not reported by the source device.

The rules for calculating and reporting jitter are the same as reporting delay (see delay discussion above)

Suggested Text for jitter (pick one or modify as necessary):

A *QosDevice* Service with a *<technology name>* interface MUST return only its contribution to the MaxCommittedJitter value for its QoS Segment. All other QosDevices on the QoS Segment MUST also return their own contribution.

A *QosDevice* Service with a *<technology name>* interface that is the source of the traffic stream in the *<technology name>* QoS Segment MUST return the sum of MaxCommittedJitter values for all the QosDevices on the QoS Segment. All other QosDevices on the QoS Segment MUST return zero or any delay value not reported by the source device.

A *QosDevice* Service with a *<technology name>* interface that is the sink of the traffic stream in the *<technology name>* QoS segment MUST return the sum of MaxCommittedJitter values for all the QosDevices on the segment. All other QosDevices on the QoS Segment MUST return zero or any delay value not reported by the sink device.

Table B.4 — AllocatedResources Parameters

UPnP-QoS parameter	*As calculated from <technology>* Parameter(s)	Comment
MaxCommittedDelay		
MaxCommittedJitter		
ListOfLayer2StreamIds		

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch