QosDevice:1

For UPnP™ Version 1.0
Date: March 10th, 2005

Authors	Company
Narm Gadiraju	Intel Corporation
Rajendra Bopardikar	Intel Corporation

Contents

List of Tables

1. Overview and Scope

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

This service-type enables modeling of 'QoS Device' function capabilities. QosDevice service is a function typically implemented in source, sink and intermediate network elements that are in the path of traffic. QosDevice Service is responsible for providing the appropriate network resources to traffic and state of the device as requested by QosManager Service. [QM]

This document does not address the procedure for end to end set up a new traffic stream or revoke an existing traffic stream.

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 section 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:1 Service Document: Note that only the schema definition used for the A_ARG_TYPE_TrafficDescriptor is normative for UPnP QosDevice:1 service specification and the schema is defined in the UPnP QosManager:1 document.

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

1.1.2. Informative References

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

[Qos Architecture] – UPnP Qos Architecture:1 Document

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:1

The shorthand QosDevice is used herein to refer to this type of service.

2.2. Namespaces

The XML[XML] in this document should be read as if the following namespace definitions are in effect.

```
xmlns="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"[QM]
xmlns="urn:schemas-upnp-org:service:QosManager:1"[QM]
```

2.3. State Variables

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

2.3.1. Derived data Types

This section defines some derived data types that are represented as UPnP string data types with special syntax.

2.3.1.1. XML Fragments as UPnP Arguments

The UPnP QoS Framework 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).

The XML schemas used in UPnP-QoS are defined in the respective files located on `http://www.upnp.org/schemas/`

In their XML fragments, implementations may use an explicit reference to appropriate name spaces.

At several places in the XML schemas there is room for vendor differentiation through the use of the "any"-tag. When extending UPnP-QoS with their own XML tags, vendors should use a name space to prevent collisions of their tags with those of other vendors. It is recommended that implementations are not required to retrieve the corresponding schemas from the Internet.

Finally, an XML fragment, in adherence to the UPnP V1.0 architecture[Qos Architecture], needs to be escaped by using the normal XML rules, [XML] Section 2.4 Character Data and Markup, before embedding it in a SOAP request or response message. The XML escaping rules are summarized from the [XML] reference mentioned above:

- The (<) character is encoded as (&lt;)
- The (>) character is encoded as (&gt;)
- The (&) character is encoded as (&amp;)
- The (") character is encoded as (&quot;)

- The (‘) character is encoded as (&apos;)

Table 2-1: State Variables

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value [2]	Default Value [2]	Eng. Units
A_ARG_TYPE_TrafficDescriptor	R	String (XML fragment)	See section 2.3.2	n/a	n/a
A_ARG_TYPE_TrafficDescriptorsPerInterface	R	String (XML fragment)	See section 2.3.3	n/a	n/a
A_ARG_TYPE_TrafficHandle	R	String	See section 2.3.4	n/a	n/a
A_ARG_TYPE_NumTraffficDescriptors	R	Integer	See section 2.3.5	n/a	n/a
A_ARG_TYPE_QosDeviceCapabilities	R	String (XML fragment)	See section 2.3.6	n/a	n/a
A_ARG_TYPE_QosDeviceState	R	String (XML fragment)	See section 2.3.7	n/a	n/a
PathInformation	O	String (XML fragment)	See section 2.3.8	n/a	n/a
A_ARG_TYPE_QosDeviceInfo	O	String (XML fragment)	See section 2.3.9	n/a	n/a
A_ARG_TYPE_QosStateId	R	String		n/a	n/a

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

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

2.3.2. A_ARG_TYPE_TrafficDescriptor

This is an escaped XML string, as specified in section 2.3.1.1, which contains QoS related information for a traffic stream. Refer to [QM] document, for details of this XML fragment using the namespace, `xmlns="http://www.upnp.org/schemas/TrafficDescriptorv1.xsd"`.

2.3.3. A_ARG_TYPE_TrafficDescriptorsPerInterface

This is an escaped XML string, as specified in section 2.3.1.1, which contains the list of traffic descriptors that are associated with a network interface on a given QosDevice.

2.3.3.1. XML Schema Definition

```
<xs:schema
xmlns="http://www.upnp.org/schemas/TrafficDescriptorsPerInterface.xsd"
  elementFormDefault="qualified"

targetNamespace="http://www.upnp.org/schemas/TrafficDescriptorsPerInterf
ace.xsd"
  xmlns ="http://www.upnp.org/schemas"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import namespace="http://www.upnp.org/schemas"
schemaLocation="TrafficDescriptorv1.xsd"/>

  <xs:annotation>
    <xs:documentation xml:lang="en">
    Traffic Descriptors Per Interface schema.
    Copyright 2004 UPnP(tm). All rights reserved.
    </xs:documentation>
  </xs:annotation>

  <xs:element name="TrafficDescriptorsPerInterface">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="TdInterfacePair" minOccurs="0"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="upnp-qos:TrafficDescriptor"
minOccurs="1" maxOccurs="1" />
              <xs:element name="InterfaceId" type="xs:string"
minOccurs="1" maxOccurs="unbounded" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
</xs:schema>
```

2.3.3.2. Description of fields in the TrafficDescriptorsPerInterface structure

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

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

InterfaceId: This is a required field. The value is of type string and is a unique value for a given device and is used to identify the interface.

2.3.3.3. Sample argument XML string

```
<TrafficDescriptorsPerInterface
xmlns="http://www.upnp.org/schemas/TrafficDescriptorsPerInterface.xsd
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <TdInterfacePair>
      <TrafficDescriptor> ...
      </TrafficDescriptor>
      <InterfaceId>eth0</InterfaceId>
    </TdInterfacePair>
    <TdInterfacePair>
      <TrafficDescriptor> ...
      </TrafficDescriptor>
      <InterfaceId>eth0</InterfaceId>
    </TdInterfacePair>
    <TdInterfacePair>
      <TrafficDescriptor> ...
      </TrafficDescriptor>
      <InterfaceId>eth0</InterfaceId>
      <InterfaceId>eth1</InterfaceId>
    </TdInterfacePair>
  <TrafficDescriptorsPerInterface>
```

2.3.4. A_ARG_TYPE_TrafficHandle

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

2.3.5. A_ARG_TYPE_NumTrafficDescriptors

This is an integer argument. Refer to the [QM] document for more details.

2.3.6. A_ARG_TYPE_QosDeviceCapabilities

This is an escaped XML fragment, as specified in section 2.3.1.1, and contains information describing a device's QoS capabilities. Refer to "`xmlns=http://www.upnp.org/schemas/QosDeviceCapabilities.xsd`" for syntactic details of this structure.

2.3.6.1. XML Schema Definition

```
<xs:schema xmlns="http://www.upnp.org/schemas/QosDeviceCapabilities.xsd"
elementFormDefault="qualified" id="QosDeviceCapabilities"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:annotation>
    <xs:documentation xml:lang="en">
    QosDeviceCapabilitiesschema.
    Copyright 2004 UPnP(tm). All rights reserved.
    </xs:documentation>
</xs:annotation>

  <xs:element name="QosDeviceCapabilities">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Interface" minOccurs="1"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element minOccurs="1" maxOccurs="1" name="InterfaceId"
type="xs:string" />
              <xs:element minOccurs="0" maxOccurs="1" name="MacAddress"
type="MacAddressType" />
             <xs:simpleType minOccurs="0" maxOccurs="1"
name="IanaTechnologyType" type="xs:Integer" />
              </xs:element>
              <xs:element minOccurs="1" maxOccurs="1"
name="AdmissionControlSupported">
<xs:simpleType name="AdmissionControlType">
                  <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="No" />
                    <xs:enumeration value="Yes" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
<xs:simpleType minoccurs="1" maxOccurs="1" name="NativeQos">
<xs:union memberTypes="xs:string BasicNativeList"/>
</xs:simpleType>
<xs:simpleType name="BasicNativeList">
                  <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="Prioritized" />
                    <xs:enumeration value="BestEffort" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element minOccurs="1" maxOccurs="1" name="MaxPhyRate"
type="xs:unsignedInt" />
            <xs:any minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
</xs:element>

  <xs:complexType name="MacAddressType">
    <xs:sequence>
      <xs:element minOccurs="6" maxOccurs="8" name="octet-n"
type="xs:byte" />
```

```
    </xs:sequence>
</xs:complexType>

</xs:schema>
```

2.3.6.2. Description of fields in the QosDeviceCapabilities structure

Interface: This is a required field and defined as an XML element. This field describes a network interface on the QosDevice. 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 a required field if a given interface has an associated MacAddress. Provides the MAC address of the Interface.

InterfaceId: This is a required field. The value is of type string and is a unique value for a given device and is used to identify the interface.

IanaTechnologyType: The IanaTechnologyType is an integer that indicates media interface type, such as 802.3 (value=6) or 802.11 (value=71). The allowed integer values for this parameter are specified in the IANA reference ifType-MIB <http://www.iana.org/assignments/ianaiftype-mib>.

AdmissionControlSupported: This is a required field. AdmissionControlSupported field indicates whether the interface on the device is capable of performing device level admission control. For this version of the specification the device must set this parameter to a value of “No”.

NativeQos: This is an optional field. Contains one of the 2 values (Prioritized, BestEffort).

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

2.3.6.3. Sample argument XML string

```
<QosDeviceCapabilities xmlns="http://www.upnp.org/schemas/
QosDeviceCapabilities.xsd">
<QosDeviceCapabilities>
  <Interface>
    <MacAddress>0212abcdef11</MacAddress>
    <InterfaceId>eth0</InterfaceId>
    <IanaTechnologyType>6</IanaTechnologyType>
    <NativeQos>Prioritized</NativeQos>
    <MaxPhyRate>100000000</MaxPhyRate>
    <AdmissionControlSupported>No</AdmissionControlSupported>
  </Interface>
  <Interface>
    <MacAddress>0212abcdef12</MacAddress>
    <InterfaceId>eth1</InterfaceId>
    <IanaTechnologyType>71</IanaTechnologyType>
    <NativeQos>Prioritized</NativeQos>
    <MaxPhyRate>3000000</MaxPhyRate>
    <AdmissionControlSupported>No</AdmissionControlSupported>
  </Interface>
  <Interface>
    <MacAddress>0212abcdef13</MacAddress>
    <InterfaceId>eth2</InterfaceId>
    <IanaTechnologyType>6</IanaTechnologyType>
    <NativeQos>BestEffort</NativeQos>
    <MaxPhyRate>5000000</MaxPhyRate>
    <AdmissionControlSupported>No</AdmissionControlSupported>
  </Interface>
</QosDeviceCapabilities>
```

2.3.7. A_ARG_TYPE_QosDeviceState

A_ARG_TYPE_QosDeviceState is a structure that provides information about a device’s current QoS state.

2.3.7.1. XML Schema Definition

```
<xs:schema xmlns="http://www.upnp.org/schemas/QosDeviceState.xsd"
elementFormDefault="qualified" id="QosDeviceState"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:annotation>
    <xs:documentation xml:lang="en">
    QosDeviceState schema.
    Copyright 2004 UPnP(tm). All rights reserved.
    </xs:documentation>
</xs:annotation>
  <xs:element name="QosDeviceState">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="QosStateId"
type="xs:string" />
        <xs:element minOccurs="1" maxOccurs="unbounded"
name="Interface">
          <xs:complexType>
            <xs:sequence>
              <xs:element minOccurs="1" maxOccurs="1" name="InterfaceId"
type="xs:string" />
              <xs:element minOccurs="0" maxOccurs="4" name="IpAddress"
type="IpAddressType"/>
              <xs:element minOccurs="1" maxOccurs="1"
name="InterfaceAvailability">
                <xs:simpleType>
                  <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="1" />
                    <xs:enumeration value="0" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
             <xs:any minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
       <xs:any minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:simpleType name="Ipv4Address">
  <xs:restriction base="xs:string">
   <xs:pattern value="(([1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-
5])\.){3}([1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])" />
  </xs:restriction>
 </xs:simpleType>

<xs:simpleType name="Ipv6Address">
```

```
  <xs:restriction base="xs:hexBinary">
   <xs:length value="32" />
  </xs:restriction>
 </xs:simpleType>

<xs:complexType name="IpAddressType">
  <xs:sequence>
   <xs:choice>
    <xs:element name="Ipv4" type="IPv4Address" />
    <xs:element name="Ipv6" type="IPv6Address" />
   </xs:choice>
   <xs:element name="PrefixLength">
    <xs:simpleType>
     <xs:restriction base="xs:positiveInteger" minOccurs="0">
      <xs:minInclusive value="1" />
      <xs:maxInclusive value="128" />
     </xs:restriction>
    </xs:simpleType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
```

2.3.7.2. Description of fields in the A_ARG_TYPE_QosDeviceState *structure*

QosStateId: This is a required field. Read theory of operation for more details as to how this parameter is used.

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

InterfaceId: This is a required field. The value is of type string and unique for a device to identify the interface uniquely.

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

InterfaceAvailability: This is a required 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.

2.3.7.3. Sample argument XML string

```
<QosDeviceState xmlns="http://www.upnp.org/schemas/QosDeviceState.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>
```

2.3.8. PathInformation

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

2.3.8.1. XML Schema Definition

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="http://www.upnp.org/schemas/PathInformation.xsd"
 targetNamespace="http://www.upnp.org/schemas/PathInformation.xsd"
 elementFormDefault="qualified">

  <xs:annotation>
    <xs:documentation xml:lang="en">
  QosDevice PathInformation schema.
  Copyright 2004 UPnP(tm). All rights reserved.
  </xs:documentation>
  </xs:annotation>
  <xs:element name="DeviceReachableMacs">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="LinkReachableMacs" minOccurs="1"
maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="LinkId" type="xs:string" minOccurs="1"
maxOccurs="1" />
            <xs:element name="BridgeId" type="xs:string" minOccurs="0"
maxOccurs="1" />
            <xs:element name="MacAddress" type="MacAddressType"
minOccurs=”0” maxOccurs="1" />
            <xs:element name="ReachableMac" type="MacAddressType"
minOccurs="0" maxOccurs="unbounded" />
            <xs:any minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    <xs:any minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  </xs:element>
</xs:schema>
```

2.3.8.2. Description of fields in **PathInformation** *structure*

LinkReachableMacs: This is a required field. A LinkReachableMacs definition 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.

LinkId: This is a required field. Its value is of type string, it must be unique within the device. It identifies the layer-2 link.

MacAddress: This is a required field when available. Provides the MAC address of the interface for an end point device.

ReachableMac: Provides the MAC address(es) of end point devices that are reachable through the link, if any.

BridgedId: Identifies the links that are bridged together. All links that have the same BridgeID are interconnected within the device such that layer-2 frames are forwarded between them.

2.3.8.3. Sample argument XML string – PC with two network interfaces

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

```
<DeviceReachableMacs xmlns="http://www.upnp.org/schemas/QoSDevice.xsd">
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <MacAddress>112233aabb03</MacAddress>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth1</LinkId>
      <MacAddress>112233aabb02</MacAddress>
  </LinkReachableMacs>
</DeviceReachableMacs>
```

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

Similar to the previous example this is an example of an end point network device with two network interfaces. However this device all forwards layer-2 frames between the two network interfaces.

```
<DeviceReachableMacs xmlns="http://www.upnp.org/schemas/QoSDevice.xsd">
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <MacAddress>112233aabb03</MacAddress>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth1</LinkId>
      <MacAddress>112233aabb02</MacAddress>
</LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <BridgeId>Bridge</BridgeId>
      <ReachableMac>112233aabb03</ReachableMac>
      <ReachableMac>112233aabb02</ReachableMac>
      <ReachableMac>112233aabb01</ReachableMac>
      <ReachableMac>112233aabb04</ReachableMac>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth1</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb05</ReachableMac>
    </LinkReachableMacs>
</DeviceReachableMacs>
```

2.3.8.5. Sample argument XML string –Four port Ethernet Switch

This is an example of a layer-2 switching device that interconnects four physical Ethernet ports. The device supports layer-2 frame forwarding between all ports.

```
<DeviceReachableMacs xmlns="http://www.upnp.org/schemas/QoSDevice.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>
```

2.3.8.6. 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 layer-2 frame forwarding between all links. This includes forwarding between wireless station or to the Ethernet interface.

```
<DeviceReachableMacs xmlns="http://www.upnp.org/schemas/QoSDevice.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>
```

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

This is an example of a bridging device with two interfaces on different network technologies. It does layer-2 forwarding of frames between wireless station interface and the wired Ethernet interface.

```
<DeviceReachableMacs xmlns="http://www.upnp.org/schemas/QoSDevice.xsd">
  <LinkReachableMacs>
      <LinkId>WL0</LinkId>
      <BridgeId>Bridge0</BridgeId>
  </LinkReachableMacs>
  <LinkReachableMacs>
      <LinkId>eth0</LinkId>
      <BridgeId>Bridge0</BridgeId>
      <ReachableMac>112233aabb04</ReachableMac>
    </LinkReachableMacs>
</DeviceReachableMacs >
```

2.3.9. A_ARG_TYPE_QosDeviceInfo

A_ARG_TYPE_QosDeviceInfo is a structure that provides port numbers and protocol information associated with a traffic stream.

2.3.9.1. XML Schema Definition

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="http://www.upnp.org/schemas/QosDeviceInfo.xsd"
 targetNamespace="http://www.upnp.org/schemas/QoDeviceInfo.xsd"
 elementFormDefault="qualified">

  <xs:annotation>
    <xs:documentation xml:lang="en">
  QoS Device Path Information schema.
  Copyright 2004 UPnP(tm). All rights reserved.
  </xs:documentation>
  </xs:annotation>

  <xs:element name="QosDeviceInfo">
        <xs:complexType>
          <xs:sequence>
                  <xs:element name="TrafficHandle" type="xs:string"
minOccurs="1" maxOccurs="1"/>
            <xs:element name="SourcePort" type="IpPortNumber"
minOccurs="0" maxOccurs="1"/>
            <xs:element name="DestinationPort" type="IpPortNumber"
minOccurs="0" maxOccurs="1" />
            <xs:element name="IpProtocol" type=" IpProtocolType"
minOccurs="1" maxOccurs="1"/>
           <xs:any minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>

 <xs:simpleType name="IpPortNumber">
  <xs:restriction base="xs:positiveInteger">
   <xs:minInclusive value="0" />
   <xs:maxInclusive value="65535" />
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="IpProtocolType">
  <xs:restriction base="xs:nonNegativeInteger">
   <xs:minInclusive value="0" />
   <xs:maxInclusive value="255" />
  </xs:restriction>
 </xs:simpleType>

  </xs:element>
</xs:schema>
```

2.3.9.2. Description of fields in **A_ARG_TYPE_QosDeviceInfo** *structure*

TrafficHandle: This is a required field that identifies the Traffic Descriptor for which QoS device information is being returned.

SourcePort: This value represents the source port that is going to be used for a traffic stream.

DestinationPort: This value represents the destination port that is going to be used for a traffic stream.

Protocol: This field represents the IANA assigned protocol number of the traffic stream.

2.3.9.3. Sample argument XML string

```
<QosDeviceInfo xmlns="http://www.upnp.org/schemas/QosDevice.xsd">
             <TrafficHandle>abcxyz</TrafficHandle>
      <SourcePort>1001</SourcePort>
      <DestinationPort>2003</DestinationPort>
      <Protocol>6</Protocol>
</QosDeviceInfo>
```

2.3.10.Relationships between State Variables

2.4. Eventing and Moderation

Table 2-2: Event Moderation

Variable Name	Evented	Moderated Event	Max Event Rate[1]	Logical Combination	Min Delta per Event[2]
PathInformation	*Yes*	*Yes*	*2*	*NA*	*NA*

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

2.4.1. Event Model

The state variable PathInformation is optional, but must be evented when implemented..

Any time there is a change in Path information, the QoS device 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.

2.5. Actions

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

Table 2-3: Actions

Name	Req. or Opt. [1]
GetQosDeviceCapabilities	R
GetQosState	R

Name	Req. or Opt. [1]
SetupTrafficQos	R
ReleaseTrafficQos	R
GetPathInformation	O
GetQosDeviceInfo	O

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

2.5.1. GetQosDeviceCapabilities

This action returns the static QoS capabilities of the QosDevice.

2.5.1.1. Arguments

Table 2-4: Arguments for GetQosDeviceCapabilities

Argument	Direction	relatedStateVariable
QosDeviceCapabilities	Out	A_ARG_TYPE_QosDeviceCapabilities

2.5.1.2. Dependency on State (if any)

None, these are static capabilities.

2.5.1.3. Effect on State (if any)

none

2.5.1.4. Errors

Refer to UPnP Device architecture for common error codes.

Table 2-5: Error Codes for GetQosDeviceCapabilities

errorCode	errorDescription	Description

2.5.2. GetQosState

This action returns the instantaneous QoS state of the device. The device must list the TrafficDescriptor(s) in the ListOfTrafficDescriptors argument in the order they were admitted to the device by the QosManager(s).

2.5.2.1. Arguments

Table 2-6: Arguments for GetQosState

Argument	Direction	relatedStateVariable
QosDeviceState	Out	A_ARG_TYPE_QosDeviceState

Argument	Direction	relatedStateVariable
NumberOfTrafficDescriptors	Out	A_ARG_TYPE_NumTrafficDescriptors
ListOfTrafficDescriptors	Out	A_ARG_TYPE_TrafficDescriptorsPerInterface

2.5.2.2. Dependency on State (if any)

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

2.5.2.3. Effect on State (if any)

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

2.5.2.4. Errors

Table 2-7: Error Codes for GetQosState

errorCode	errorDescription	Description

2.5.3. SetupTrafficQos

SetupTrafficQoS interface indicates to the device to setup QoS for the Traffic described by A_ARG_TYPE_TrafficDescriptor. If there is no traffic descriptor with the same A_ARG_TYPE_TrafficHandle, then the traffic descriptor is registered in the device. If the device already has the traffic descriptor (identified by the traffic handle) registered, then the device must return an error (Error Code 702).

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

Typically, the QoS Manager calls this action only once per traffic handle registration. If the QoS Manager intends to update QoS associated with the traffic (e.g. the lease time of the traffic), then it has to go over the complete traffic setup process again after it has released the QoS.

2.5.3.1. Arguments

Table 2-8: Arguments for SetupTrafficQos

Argument	Direction	relatedStateVariable
SetupTrafficDescriptor	In	A_ARG_TYPE_TrafficDescriptor
QosStateId	In	A_ARG_TYPE_QosStateId

2.5.3.2. Dependency on State (if any)

QosStateId is provided as as input to this action. In case the current QosStateId of the device is different than the one specified by the QosManager, the action returns an error (Error Code 760). Otherwise, the QosDevice sets up QoS for the traffic.

2.5.3.3. Effect on State (if any)

Upon successful completion of this action, the QosDevice sets up QoS for the traffic specified in the action request. Please refer to 'Theory of Operation' section for more details.

2.5.3.4. Errors

Table 2-9: Error Codes for SetupTrafficQos

errorCode	errorDescription	Description
700	Traffic Handle missing or empty	Traffic Handle must be filled in as input to this action.
702	Traffic Handle already registered	A QosManager is not allowed to setup or modify QoS using SetupTrafficQos if QoS has already been setup for that handle.
710	Incomplete TrafficId	All TrafficId fields (Source Ip, Destination IP, Source Port, Destination Port and Protocol) must be present.
711	Insufficient information	The input information is not complete.
720	ActiveTspecIndex is not a TspecIndex	
751	Device not on path	
760	QosStateId does not match	Please refer to the 'Theory of Operation' section.
761	QosDevice cannot setup this stream	QoS Setup failed, e.g device does not support prioritized QoS

2.5.4. ReleaseTrafficQos

ReleaseTrafficQoS provides an indication to the device to release the QoS for the traffic identified by A_ARG_TYPE_TrafficHandle. After this call, traffic handle is no longer registered at the device to provide QoS. The device must release all its QoS resources allocated to that traffic.

2.5.4.1. Arguments

Table 2-10: Arguments for ReleaseTrafficQos

Argument	Direction	relatedStateVariable
ReleaseTrafficHandle	In	A_ARG_TYPE_TrafficHandle

2.5.4.2. Dependency on State (if any)

The TrafficHandle provided has to be valid.

2.5.4.3. Effect on State (if any)

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

2.5.4.4. Errors

Table 2-11: Error Codes for ReleaseTrafficQos

errorCode	errorDescription	Description
703	Traffic Handle unknown to this device	

2.5.5. GetPathInformation

This is an optional action. When supported, this action call returns the 'PathInformation' structure for that QosDevice service providing information about the reachable MACs. This information may be used by the QosManager for path determination.

2.5.5.1. Arguments

Table 2-12: Arguments for GetPathInformation

Argument	Direction	relatedStateVariable
PathInformation	Out	PathInformation

2.5.5.2. Dependency on State (if any)

None.

2.5.5.3. Effect on State (if any)

None.

2.5.5.4. Errors

Table 2-13: Error Codes for GetPathInformation

errorCode	errorDescription	Description

2.5.6. GetQosDeviceInfo

This is an optional action. When supported, this action call returns the 'QosDeviceInfo' structure for that QosDevice service providing information about the port number and protocol information associated with the provided TrafficDescriptor. The device may be able to determine this information based on the following:

- Available elements of the TrafficId
- AvTransportUri and AvTransportInstanceId if specified
- MediaServerConnectionId and MediaRendererConnectionId if specified

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

2.5.6.1. Arguments

Table 2-14: Arguments for GetQosDeviceInfo

Argument	Direction	relatedStateVariable
TrafficDescriptor	In	A_ARG_TYPE_TrafficDescriptor
QosDeviceInfo	Out	A_ARG_TYPE_QosDeviceInfo

2.5.6.2. Dependency on State (if any)

2.5.6.3. Effect on State (if any)

None.

2.5.6.4. Errors

Table 2-15: 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.

2.5.7. 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 section on Description).

2.5.8. Relationships Between Actions

2.5.9. Common Error Codes

The following table lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error must be returned. These common error codes are defined in the UPnP™ Device Architecture [DEVICE] and other Technical Committee documents.

Table 2-16: Common Error Codes

errorCode	errorDescription	Description
400-499	TBD	See UPnP™ Device Architecture section on Control.
500-599	TBD	See UPnP™ Device Architecture section on Control.
600-699	TBD	See UPnP™ Device Architecture section on Control.
700	Traffic Handle missing or empty	Traffic Handle must be filled in as input to this action.
702	Traffic Handle already registered	A QosManager is not allowed to setup or modify QoS using SetupTrafficQos if QoS has already been setup for that handle.

errorCode	errorDescription	Description
703	Traffic Handle unknown to this device	
710	Incomplete TrafficId	All TrafficId fields (Source Ip, Destination IP, Source Port, Destination Port and Protocol) 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.
720	ActiveTspecIndex is not a TspecIndex	
751	Device not on path	
760	QosStateId does not match	Please refer to the 'Theory of Operation' section.
761	QosDevice cannot setup this stream	QoS Setup failed, e.g device does not support prioritized QoS
800-899	*TBD*	*(Specified by UPnP™ vendor.)*

2.6. Theory of Operation

The QosDevice Service provides an action for control points to query the QoS state of the device, execute actions on the device and register for events the device generates. Typically the QoS Management Entity interacts with the QosDevice Service.

A UPnP QosDevice will expose 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. Although currently device level admission control is not supported, there is an element 'AdmissionControlSupported' which is expected to always return the value of 'No'.

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 QoS Management Entity to request the device to provide network resources for a traffic stream identified by the TrafficDescriptor.

Race conditions may occur when different QoS Management Entities use the actions GetQosState and SetupTrafficQos. To identify such conditions, the QosDevice provides a unique identification of its state named QosStateId in reply to the action GetQosState. The QoS Management Entity provides QosStateId as input argument to SetupTrafficQos. In case the QosStateId sent by the QoS Management Entity does not match the most recent QosStateId handed out by the device, the device responds to SetupTrafficQoS with error code 760.

The QoS Management Entity can indicate to the QosDevice to release resources in two ways. The QoS Management Entity may either call the ReleaseTrafficQoS action or simply not renew the lease time associated with the traffic stream.

The QosDevice provides layer 2 reachability information to the QoS Management Entity through the GetPathInformation action. This infornation includes the link identifier, MAC address, brdidging information and reachable MAC addresses for each link on the QosDevice. This is useful to help a QoS Management Entity determine the topology of the network and the path of the traffic streams.

A QoS Management Entity may provide a lease time as part of a TrafficDescriptor as input to SetupTrafficQos. When the lease time expires, the QosDevice releases the QoS resources allocated to that TrafficDescriptor. In case, QoS Management Entity does not specify the lease time, the QoS resources remain allocated until they are released by a QoS Management Entity.

The persistence of any QosDevice information across reboots is not required.

3. XML Service Descriptions

```
<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-upnp-org:QosDevice:1">

   <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>GetQosDeviceCapabilities</name>
      <argumentList>
         <argument>
            <name>QosDeviceCapabilities</name>
            <direction>out</direction>

<relatedStateVariable>A_ARG_TYPE_QosDeviceCapabilities</relatedStateVari
able>
         </argument>
      </argumentList>
   </action>
   <action>
      <name>GetQosDeviceInfo</name>
      <argumentList>
         <argument>
            <name>SetupTrafficDescriptor</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</relatedStateVari
able>
         </argument>
         <argument>
            <name>ListOfTrafficDescriptors</name>
            <direction>out</direction>

<relatedStateVariable>A_ARG_TYPE_TrafficDescriptorsPerInterface</related
StateVariable>
         </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>
</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>
</serviceStateTable>
</scpd>
```

4. Test

No semantic tests have been specified for this service.