IETF Internet Draft NSIS Working Group
Internet Draft
<draft-ietf-nsis-qspec-12.txt>
Expiration Date: April 2007

Jerry Ash
AT&T
Attila Bader
Ericsson
Cornelia Kappler
Siemens AG

October 2006

QoS NSLP QSPEC Template

Status of this Memo

Copyright Notice

Abstract

The QoS NSLP protocol is used to signal QoS reservations and is independent of a specific QoS model (QOSM) such as IntServ or DiffServ. Rather, all information specific to a QOSM is encapsulated in a separate object, the QSPEC. This document defines a template for the QSPEC, which contains both the QoS description and QSPEC control information. The QSPEC format is defined, as are a number of QSPEC parameters. The QSPEC-1 parameters provide a common language

to be re-used in several QOSMs. QSPEC-2 parameters aim to ensure

the extensibility of QoS NSLP to other QOSMs in the future. To a certain extent QSPEC parameters ensure interoperability of QOSMs. The node initiating the NSIS signaling adds an Initiator QSPEC that must not be removed, thereby ensuring the intention of the NSIS initiator is preserved along the signaling path.

Table of Contents

Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1. Contributors

This document is the result of the NSIS Working Group effort. In addition to the authors/editors listed in Section 13, the following people contributed to the document:

Chuck Dvorak
AT&T
Room 2A37
180 Park Avenue, Building 2
Florham Park, NJ 07932

Phone: + 1 973-236-6700
Fax:+1 973-236-7453
Email: cdvorak@att.com

Yacine El Mghazli
Alcatel
Route de Nozay
91460 Marcoussis cedex
FRANCE
Phone: +33 1 69 63 41 87
Email: yacine.el_mghazli@alcatel.fr

Georgios Karagiannis
University of Twente
P.O. BOX 217
7500 AE Enschede
The Netherlands
Email: g.karagiannis@ewi.utwente.nl

Andrew McDonald
Siemens/Roke Manor Research
Roke Manor Research Ltd.
Romsey, Hants SO51 0ZN
UK
Email: andrew.mcdonald@roke.co.uk

Al Morton
AT&T
Room D3-3C06
200 S. Laurel Avenue
Middletown, NJ 07748
Phone: + 1 732 420-1571
Fax: +.1 732 368-1192
Email: acmorton@att.com

Percy Tarapore
AT&T
Room D1-33
200 S. Laurel Avenue
Middletown, NJ 07748
Phone: + 1 732 420-4172
Email: tarapore@.att.com

Lars Westberg
Ericsson Research
Torshamnsgatan 23
SE-164 80 Stockholm, Sweden
Email: Lars.Westberg@ericsson.com

2. Introduction

The QoS NSIS signaling layer protocol (NSLP) [QoS-SIG] establishes and maintains state at nodes along the path of a data flow for the purpose of providing forwarding resources (QoS) for that flow. The design of QoS NSLP is conceptually similar to RSVP [RFC2205], and meets the requirements of [RFC3726].

A QoS-enabled domain supports a particular QoS model (QOSM), which is a method to achieve QoS for a traffic flow. A QOSM incorporates QoS provisioning methods and a QoS architecture. It defines the behavior of the resource management function (RMF) defined in [QoS-SIG], including inputs and outputs. Examples of QOSMs are IntServ, DiffServ admission control, and those specified in [Y.1541-QOSM, CL-QOSM, RMD-QOSM].

The QoS NSLP protocol is used to signal QoS reservations and supports signaling for different QOSMs. All information specific to a QOSM is encapsulated in the QoS specification (QSPEC) object, which is QOSM specific, and this document defines a template for the QSPEC.

Since QoS NSLP signaling operation can be different for different QOSMs, the QSPEC contains two kinds of information, QSPEC control information and QoS description. QSPEC control information contains parameters that governs the behavior of the RMF. An example of QSPEC control information is how the excess traffic is treated in the RMF queuing functions. The QoS description parameters include, for example, <Traffic> parameters, such as <Token Bucket> and <Bandwidth>, and constraints parameters, such as <PHB Class> and <Path Latency>.

The QoS description is composed of QSPEC objects loosely corresponding to the TSpec, RSpec and AdSpec objects specified in RSVP. This is, the QSPEC may contain a description of QoS desired and QoS reserved. It can also collect information about available resources. Going beyond RSVP functionality, the QoS description also allows indicating a range of acceptable QoS by defining a QSPEC object denoting minimum QoS. Usage of these QSPEC objects is not bound to particular message types thus allowing for flexibility. A QSPEC object collecting information about available resources may travel in any QoS NSLP message, for example a QUERY message or a RESERVE message. The QSPEC travels in QoS NSLP messages but is opaque to the QoS NSLP, and is only interpreted by the RMF.

Interoperability between QoS NSIS entities (QNEs) in different domains that implement different QOSMs is enhanced (but not guaranteed) by the definition of a common set of QSPEC-1 and QSPEC-2 parameters. QSPEC-1 parameters in the QSPEC must be interpreted by all QNEs in the path, independent of which QOSM they support. This way, NSIS provides a mechanism for interdomain QoS signaling and interworking. QSPEC-2 parameters, in contrast, may be

skipped if not understood. Additional QSPEC-2 parameters can be defined by QOSM specification documents, and thereby ensure the extensibility and flexibility of QoS NSLP.

A QoS NSIS initiator (QNI) initiating the QoS NSLP signaling adds an initiator QSPEC object containing parameters describing the desired QoS based on the QOSM it supports. A local QSPEC can be stacked on the initiator QSPEC to accommodate different QOSMs being used in different domains. A domain supporting a different local QOSM than the QNI can interpret the initiator QSPEC and stack a local QSPEC to meet the local QOSM requirements. If the local domain cannot fully interpret the initiator QSPEC, it can flag the condition and

either continue to forward the reservation or possibly reject the reservation.

Thus, one of the major differences between RSVP and QoS NSLP is that QoS NSLP supports signaling for different QOSMs along the data path, all with one signaling message. For example, the data path may start in a domain supporting DiffServ and end in a domain supporting Y.1541. The ability to achieve end-to-end QoS through multiple Internet domains is also an important requirement, and illustrated in this document.

3. Terminology

Minimum QoS: Minimum QoS is a QSPEC object that MAY be supported by any QNE. Together with a description of QoS Desired or QoS Available, it allows the QNI to specify a QoS range, i.e. an upper and lower bound. If the QoS Desired cannot be reserved, QNEs are going to decrease the reservation until the minimum QoS is hit.

QNE: QoS NSIS Entity, a node supporting QoS NSLP.

QNI: QoS NSIS Initiator, a node initiating QoS NSLP signaling.

QNR: QoS NSIS Receiver, a node terminating QoS NSLP signaling.

QoS Description: Describes the actual QoS in QSPEC objects QoS Desired, QoS Available, QoS Reserved, and Minimum QoS. These QSPEC objects are input or output parameters of the RMF. In a valid QSPEC, at least one QSPEC object of the type QoS Desired, QoS Available or QoS Reserved MUST be included.

QoS Available: QSPEC object containing parameters describing the available resources. They are used to collect information along a reservation path.

QoS Desired: QSPEC object containing parameters describing the desired QoS for which the sender requests reservation.

QoS Model (QOSM): A method to achieve QoS for a traffic flow, e.g.,

IntServ Controlled Load. A QOSM specifies a set of QSPEC-1 and QSPEC-2 parameters that describe the QoS and how resources will be managed by the RMF. It furthermore specifies how to use QoS NSLP to signal for this QOSM.

QoS Reserved: QSPEC object containing parameters describing the reserved resources and related QoS parameters, for example, bandwidth.

QSPEC Control Information: Control information that is specific to a QSPEC, and contains parameters that govern the RMF.

QSPEC: QSPEC is the object of QoS NSLP containing all QOSM-specific information.

QSPEC parameter: Any parameter appearing in a QSPEC; includes both QoS description and QSPEC control information parameters, for example, bandwidth, token bucket, and excess treatment parameters.

QSPEC Object: Main building blocks of QoS Description containing a QSPEC parameter set that is input or output of an RMF operation.

QSPEC-1 parameter: QSPEC parameter that a QNI SHOULD populate if applicable to the QOSM supported by the QNI and a QNE MUST interpret, if populated.

QSPEC-2 parameter: QSPEC parameter that a QNI SHOULD populate if applicable to the QOSM supported by the QNI, and a QNE SHOULD interpret if populated and applicable to the QOSM(s) supported by the QNE. (A QNE MAY ignore if it does not support a QOSM needing the QSPEC-2 parameter).

Resource Management Function (RMF): Functions that are related to resource management, specific to a QOSM. It processes the QoS description parameters and QSPEC control parameters.

Read-only Parameter: QSPEC Parameter that is set by initiating or responding QNE and is not changed during the processing of the QSPEC along the path.

Read-write Parameter: QSPEC Parameter that can be changed during the processing of the QSPEC by any QNE along the path.

4. QOSM/NSLP Framework

The overall framework for the QoS NSLP is that [QoS-SIG] defines QoS signaling and semantics, the QSPEC template defines the container and semantics for QoS parameters and objects, and QOSM specifications define QoS methods and procedures for using QoS signaling and QSPEC parameters/objects within specific QoS deployments. QoS NSLP is a generic QoS signaling protocol that can signal for many QOSMs.

A QOSM is a method to achieve QoS for a traffic flow, e.g., IntServ controlled load [CL-QOSM], resource management with DiffServ [RMD-QOSM], and QoS signaling for Y.1541 QoS classes [Y.1541-QOSM]. A QOSM specifies a set of QSPEC parameters that describe the QoS desired and how resources will be managed by the RMF. It furthermore specifies how to use QoS NSLP to signal for this QOSM. The QSPEC is the object of QoS NSLP containing all QOSM-specific information, which is described in the next section, such as QoS description parameters (e.g., bandwidth) and QSPEC control information parameters (e.g., excess treatment). The RMF implements functions that are related to resource management, specific to a QOSM and processes the QSPEC parameters.

The QOSM specification includes how the requested QoS resources will be described and how they will be managed by the RMF. For this purpose, the QOSM specification defines a set of QSPEC-1 and QSPEC-2 parameters it uses to describe the desired QoS and QoS resource control in the RMF, and it may define additional QSPEC-2 parameters. QSPEC-1 parameters are populated by a QNI if they are applicable to the underlying QOSM the QNI supports and that a QNE must interpret, if populated. QSPEC-2 parameters are populated by a QNI if they are applicable to the underlying QOSM a QNI supports, and a QNE should interpret if populated and applicable to the QOSM(s) supported by the QNE.

A QNE MUST support at least one QOSM. A QoS-enabled domain supports a particular QOSM, and the QNEs in the domain MUST also support the QOSM.

A QOSM specification MUST include the following:

- role of QNEs, e.g., location, frequency, statefulness, etc.
- QSPEC definition including QOSM ID, QSPEC parameters
- QSPEC procedures applicable to this QOSM
- QNE processing rules describing how QSPEC information is treated and interpreted in the RMF and QOSM specific processing. E.g., admission control, scheduling, policy control, QoS parameter accumulation (e.g., delay).
- at least one bit-level QSPEC example
- QSPEC parameter behavior for new QSPEC-2 parameters the QOSM specification defines
- QSPEC parameter behavior for remapping of existing QSPEC parameters, as described in Section 6.3.2. Remapping may result in slight modification to the intended specification when strict interpretation is not possible. Unless otherwise specified in the QOSM specification document, the default QOSM behaviors for all

QSPEC-1 parameters is to strictly interpret the QSPEC-1 parameters as defined in this document through the references that precisely define the QSPEC parameter behaviors.
- define what happens in case of preemption if the default QNI

behavior (tear down preempted reservation) is not followed (see Section 6.3.5)

A QOSM specification MAY include the following:

- QOSM-specific control information parameters and processing rules for those parameters
- define additional QOSM-specific error codes, as discussed in Section 6.3.4
- specify the conditions for rejecting a reservation when the non-QOSM-hop Q-flag and remapped QSPEC parameter R-flags are sent back in the RESPONSE message (in the absence of such procedures, the default condition is 'success' if all QSPEC parameters are met and 'reservation failure' if one or more QSPEC parameters are not met)

5. QSPEC Framework

The QSPEC is the object of QoS NSLP containing all QOSM-specific information, and contains QSPEC objects and parameters. QSPEC objects are the main building blocks of the QoS description containing a QSPEC parameter set that is input or output of an RMF operation. QSPEC parameters are the parameters appearing in a QSPEC, which include both QoS description parameters (e.g., bandwidth) and QSPEC control information parameters (e.g., excess treatment). The RMF implements functions that are related to resource management, specific to a QOSM. It processes the QoS description parameters and QSPEC control information parameters.

QSPEC-1 parameters provide a common language for QOSM developers to build their QSPECs and are likely to be re-used in several QOSMs. QSPEC-1 parameters are populated by a QNI if they are applicable to the underlying QOSM the QNI supports and that a QNE must interpret, if populated. QSPEC-2 parameters are populated by a QNI if they are applicable to the underlying QOSM a QNI supports, and a QNE should interpret if populated and applicable to the QOSM(s) supported by the QNE. Note that a QNE may ignore a QSPEC-2 parameter if it does not support a QOSM needing the QSPEC-2 parameter. QSPEC-1 and QSPEC-2 parameters are defined in this document, and additional QSPEC-2

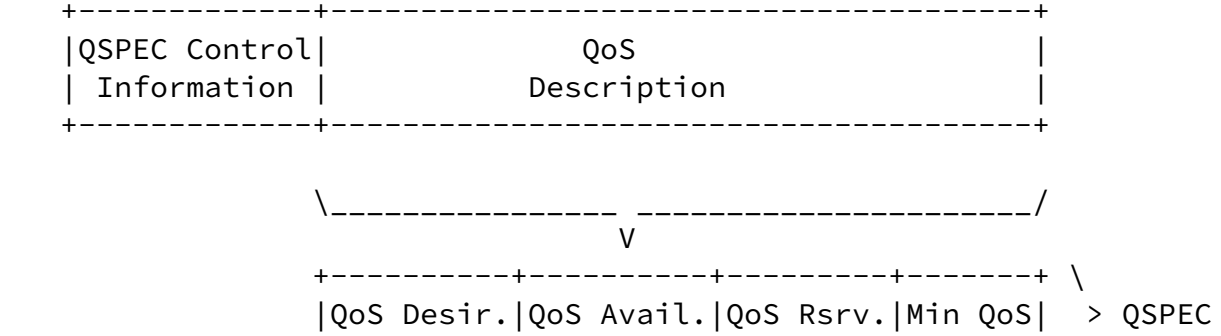
parameters may be defined in separate QOSM specification documents. For example, QSPEC-2 parameters are defined in [RMD-QOSM] and [Y.1541-QOSM]. The set of QSPEC-1 parameters in NSIS is based on DiffServ and IntServ/RSVP. Note that in effect all parameters are QSPEC-1 in RSVP since it does not have the QSPEC-1/QSPEC-2 concept.

In this document the term 'interpret' means, in relation to RMF processing of QSPEC parameters, either that the RMF a) strictly interprets a QSPEC parameter, or b) remaps, approximates, or otherwise does not strictly interpret the parameter. Furthermore, the terminology 'strictly interpret' means that the QSPEC parameter is processed by the RMF according to the commonly accepted normative

procedures specified by references given for each QSPEC parameter. Otherwise the QSPEC parameter may be remapped or approximately interpreted. For example a token bucket parameter may be remapped to bandwidth and simply interpreted by the RMF as bandwidth. Note also that a QNE must interpret a QSPEC-1 parameter only if it is populated in the QSPEC object by the QNI. If a QSPEC-1 parameter is not there in the QSPEC, the QNE does not interpret it of course. To test compliance, however, a QNE would need to be tested that it properly implements/interprets all QSPEC-1 parameters.

5.1 QSPEC Objects

This document provides a template for the QSPEC, which is needed in order to help define individual QOSMs and in order to promote interoperability between QOSMs. Figure 1 illustrates how the QSPEC is composed of QSPEC control information and QoS description. QoS description in turn is composed of up to four QSPEC objects (not all of them need to be present), namely QoS Desired, QoS Available, QoS Reserved and Minimum QoS. Each of these QSPEC Objects, as well as QSPEC Control Information, consists of a number of QSPEC-1 and QSPEC-2 parameters.



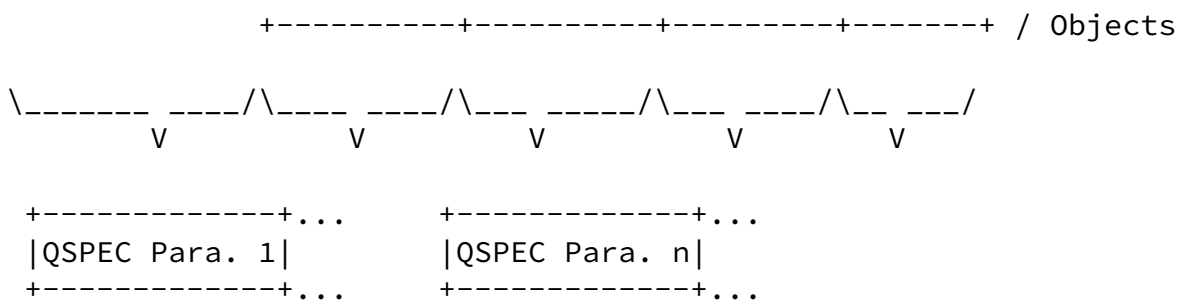


Figure 1: Structure of the QSPEC

The internal structure of each QSPEC object and the QSPEC control information, with QSPEC-1 and QSPEC-2 parameters, is illustrated in Figure 2.

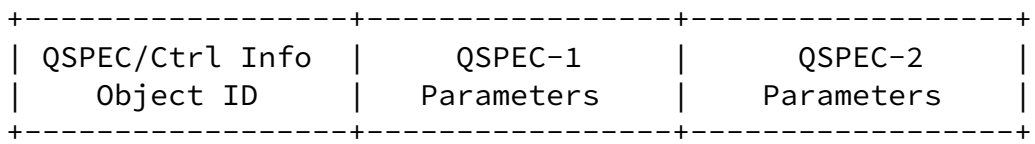


Figure 2: Structure of QSPEC Objects & Control Information

5.2 QSPEC Parameters

5.2.1 QSPEC-1 and QSPEC-2 Parameters

QSPEC-1 and QSPEC-2 parameters are defined in this document and are applicable to a number of QOSMs. QSPEC-1 parameters are treated as follows:

- o A QNI SHOULD populate QSPEC-1 parameters if applicable to the QOSM supported by the QNI.
- o QNEs/QNR MUST interpret QSPEC-1 parameters, if signaled.

QSPEC-2 parameters are treated as follows:

- o A QNI SHOULD populate QSPEC-2 parameters if applicable to the QOSM for which it is signaling.

- o QNEs/QNR SHOULD interpret QSPEC-2 parameters, if signaled and applicable to the QOSM(s) supported by the QNE/QNR. (A QNE/QNR MAY ignore the QSPEC-2 parameter if it does not support a QOSM needing the QSPEC-2 parameter).

Note that a QNE that stacks 2 QSPECs should follow the same rules as a QNI. That is, when there are two stacked QSPECs in a local domain, the QNEs in the interior local domain need only process the local (topmost) QSPEC and can ignore the initiator (bottom) QSPEC. However, edge QNEs in the local domain indeed must interpret the QSPEC-1 parameters populated in the initiator QSPEC.

This specification defines 4 QSPEC-1 parameters: <Excess Treatment>, <Traffic>, <QoS Class>, and <Priority>. The coding for these parameters is specified in Section 7.2. Due to a lack of sufficient deployment experience this is a best guess, which should be reviewed once operational experience requires or allows such a review. This specification also defines 10 QSPEC-2 parameters, and the coding for these parameters is specified in Section 7.3.

5.2.2 Read-only and Read-write QSPEC Parameters

QoS description parameters can be either read-only or read-write, depending on which QSPEC object, and which message, they appear in. In particular, all parameters in the QoS Desired object, QoS Reserved object, and Minimum QoS object are read-only for all messages. Parameters in the QoS Available object are normally read-write

---

parameters when the QoS Available object appears for the first time e.g. in the RESERVE message or QUERY message from QNI to QNR. However, on its way back, all parameters in the <QoS Available> object are read-only, e.g., in the RESPONSE message or RESERVE message from QNR to QNI. This is because on its way back the QoS Available object just transports the information it collected before.

In the QSPEC control information object, the property of being read-write or read-only is parameter specific. Note that the only control information parameter specified in this document is the <excess treatment> parameter, which is a read-only parameter.

5.3 QSPEC Formats

```
QSPEC = <QSPEC Version> <QOSM ID> <QSPEC Control Information>
        <QoS Description>
```

As described above, the QSPEC contains an identifier for the QOSM, the actual resource description (QoS description) as well as QSPEC control information. QSPEC-1 parameters defined in the following

sections include <Excess Treatment>, <Traffic>, <QoS Class>, and <Priority>. All other QSPEC parameters defined in the following sections are QSPEC-2 parameters.

A QSPEC object ID identifies whether the object is <QSPEC Control Information> or <QoS Description>. As described below, the <QoS Description> is further broken down into <QoS Desired>, <QoS Available>, <QoS Reserved>, and <Minimum QoS> objects. A QSPEC parameter ID is assigned to identify each QSPEC parameter defined below.

<QSPEC Version> identifies the QSPEC version number. Later QSPEC versions MUST be backward compatible with earlier QSPEC versions. That is, a version n+1 device must support a version n (or earlier) QSPEC and QSPEC parameters. If the version n device receives QSPEC-1 parameters (with the M-flag set, as defined in Section 7) that are not supported in version n (only supported in version n+1), then the version n device concludes that either a) the M-flag is set incorrectly for an QSPEC-2 parameter it does not support, or b) the M-flag is correctly set for a QSPEC-1 parameter it does not support. In either case, the version n device responds with a 'Malformed QSPEC' error code (0x03), as discussed in Section 6.3.1.

A new QSPEC version MUST be defined whenever this document is reissued, for example, whenever a new QSPEC-1 parameter is added. QSPEC-1 parameters in a new QSPEC version MUST be a superset of those in the previous QSPEC version.

The <QOSM ID> identifies the particular QOSM being used by the QNI and tells a QNE which parameters to expect. This may simplify processing and error analysis. Furthermore, it may be helpful for a

QNE or a domain supporting more than one QOSM to learn which QOSM the QNI would like to have in order to use the most suitable QOSM. Even if a QNE does not support the QOSM it MUST interpret at least the QSPEC-1 parameters. Note that more parameters than required by the QOSM can be included by the QNI. QSPEC version and QOSM IDs are assigned by IANA.

5.3.1 QSPEC Control Information

QSPEC control information is used for signaling QOSM RMF functions not defined in QoS NSLP. It enables building new RMF functions required by a QOSM within a QoS NSLP signaling framework, such as

specified, for example, in [RMD-QOSM] and [Y.1541-QOSM].

<QSPEC Control Information> = <Excess Treatment>

The <Excess Treatment> parameter describes how the QNE will process excess traffic, that is, out-of-profile traffic. Excess traffic MAY be dropped, shaped and/or remarked. The excess treatment parameter is initially set by the QNI and is read-only.

5.3.2 QoS Description

The QoS Description is broken down into the following QSPEC objects:

<QoS Description> = <QoS Desired> <QoS Available> <QoS Reserved>
<Minimum QoS>

Any subset of the QSPEC objects on the right hand side of the equal sign can be included in the QSPEC. Of these QSPEC objects, QoS Desired, QoS Available and QoS Reserved MUST be supported by QNEs. Minimum QoS MAY be supported.

5.3.2.1 <QoS Desired>

<QoS Desired> = <Traffic> <QoS Class> <Priority>
<Path Latency> <Path Jitter> <Path PLR> <Path PER>

Any subset of the QSPEC parameters on the right hand side of the equal sign can be included in the <QoS desired> object. These parameters describe the resources the QNI desires to reserve and hence this is a read-only QSPEC object. The <QoS Desired> resources that the QNI wishes to reserve are of course directly related to the traffic the QNI is going to inject into the network. Therefore, when used in the <QoS Desired> object, <Traffic> refers to traffic injected by the QNI into the network.

<Traffic> = <Bandwidth> <Token Bucket>

Either sub-parameter on the right hand side of the equal sign can be included in the <Traffic> parameter. Here

<Bandwidth> = link bandwidth needed by flow [RFC2212, RFC2215]

<Token Bucket> = <r> <b> <p> <m> <MTU> [RFC2210]

All of the sub-parameters on the right hand side of the equal sign MUST be included in the <Token Bucket> parameter. Note that the Path MTU Discovery (PMTUD) working group is currently specifying a robust method for determining the MTU supported over an end-to-end path. This new method is expected to update RFC1191 and RFC1981, the current standards track protocols for this purpose.

<QoS Class> = <PHB Class> <DSTE Class Type> <Y.1541 QoS Class>

Any one of the sub-parameter on the right hand side of the equal sign can be included in the <QoS Class> parameter.

An application MAY like to reserve resources for packets with a particular QoS class, e.g. a DiffServ per-hop behavior (PHB) [RFC2475], DiffServ-aware MPLS traffic engineering (DSTE) class type [RFC3564, RFC4124], or Y.1541 QoS class [Y.1541].

<Priority> = <Preemption Priority> <Defending Priority>
<Admission Priority> <RPH Priority>

Any subset of the sub-parameter on the right hand side of the equal sign can be included in the <Priority> parameter.

<Preemption Priority> is the priority of the new flow compared with the defending priority of previously admitted flows. Once a flow is admitted, the preemption priority becomes irrelevant. <Defending Priority> is used to compare with the preemption priority of new flows. For any specific flow, its preemption priority MUST always be less than or equal to the defending priority. <Admission Priority> and <RPH Priority> provide an essential way to differentiate flows for emergency services, ETS, E911, etc., and assign them a higher admission priority than normal priority flows and best-effort priority flows.

Appropriate security measures need to be taken to prevent abuse of the <Priority> parameters, see Section 8 on Security Considerations.

[Y.1540] defines packet transfer outcomes, as follows:

Successful: packet arrives within the preset waiting time with no errors

Lost: packet fails to arrive within the waiting time

Errored: packet arrives in time, but has one or more bit errors in the header or payload

Packet Loss Ratio (PLR) = total packets lost/total packets sent

Packet Error Ratio (PER) = total errored packets/total packets sent

<Path Latency>, <Path Jitter>, <Path PLR>, and <Path PER> are QSPEC-2 parameters describing the desired path latency, path jitter and path bit error rate respectively. Since these parameters are cumulative, an individual QNE cannot decide whether the desired path latency, etc., is available, and hence they cannot decide whether a reservation fails. Rather, when these parameters are included in <Desired QoS>, the QNI SHOULD also include corresponding parameters in a <QoS Available> QSPEC object in order to facilitate collecting this information.

5.3.2.2 <QoS Available>

<QoS Available> = <Traffic> <QoS Class> <Priority>
<Path Latency> <Path Jitter> <Path PLR> <Path PER>
<Ctot> <Dtot> <Csum> <Dsum>

Any subset of the QSPEC parameters on the right hand side of the equal sign can be included in the <QoS Available> object.

When used in the <QoS Available> object, <Traffic> refers to traffic resources available at a QNE in the network.

The <QoS Available> Object collects information on the resources currently available on the path when it travels in a RESERVE or QUERY message and hence in this case this QSPEC object is read-write. Each QNE MUST inspect all parameters of this QSPEC object, and if resources available to this QNE are less than what a particular parameter says currently, the QNE MUST adapt this parameter accordingly. Hence when the message arrives at the recipient of the message, <QoS Available> reflects the bottleneck of the resources currently available on a path. It can be used in a QUERY message, for example, to collect the available resources along a data path.

When <QoS Available> travels in a RESPONSE message, it in fact just transports the result of a previous measurement performed by a RESERVE or QUERY message back to the initiator. Therefore in this case, <QoS Available> is read-only.

The parameters <Token Bucket> and <Bandwidth> provide information, for example, about the bandwidth available along the path followed by a data flow. The local parameter is an estimate of the bandwidth the QNE has available for packets following the path. Computation of the

value of this parameter SHOULD take into account all information available to the QNE about the path, taking into consideration administrative and policy controls on bandwidth, as well as physical resources. The composition rule for this parameter is the MIN

function. The composed value is the minimum of the QNE's value and the previously composed value. This quantity, when composed end-to-end, informs the QNR (or QNI in a RESPONSE message) of the minimal bandwidth link along the path from QNI to QNR.

The <Path Latency> parameter accumulates the latency of the packet forwarding process associated with each QNE, where the latency is defined to be the mean packet delay added by each QNE. This delay results from speed-of-light propagation delay, from packet processing limitations, or both. The mean delay reflects the variable queuing delay that may be present. Each QNE MUST add the propagation delay of its outgoing link, if this link exists. Furthermore, the QNI MUST add the propagation delay of the ingress link, if this link exists. The composition rule for the <Path Latency> parameter is summation with a clamp of (2**32 - 1) on the maximum value. This quantity, when composed end-to-end, informs the QNR (or QNI in a RESPONSE message) of the minimal packet delay along the path from QNI to QNR. The purpose of this parameter is to provide a minimum path latency for use with services which provide estimates or bounds on additional path delay [RFC2212].

The <Path Jitter> parameter accumulates the jitter of the packet forwarding process associated with each QNE, where the jitter is defined to be the nominal jitter added by each QNE. IP packet jitter, or delay variation, is defined in [RFC3393], Section 3.4 (Type-P-One-way-ipdv), and where the selection function includes the packet with minimum delay such that the distribution is equivalent to 2-point delay variation in [Y.1540]. The suggested evaluation interval is 1 minute. This jitter results from packet processing limitations, and includes any variable queuing delay which may be present. Each QNE MUST add the jitter of its outgoing link, if this link exists. Furthermore, the QNI MUST add the jitter of the ingress link, if this link exists. The composition method for the <Path Jitter> parameter is the combination of several statistics describing the delay variation distribution with a clamp on the maximum value (note that the methods of accumulation and estimation of nominal QNE jitter are specified in clause 8 of [Y.1541]). This quantity, when composed end-to-end, informs the QNR (or QNI in a RESPONSE message) of the nominal packet jitter along the path from QNI to QNR. The

purpose of this parameter is to provide a nominal path jitter for use with services that provide estimates or bounds on additional path delay [RFC2212].

The <Path PLR> parameter accumulates the packet loss rate (PLR) of the packet forwarding process associated with each QNE, where the PLR is defined to be the PLR added by each QNE. Each QNE MUST add the PLR of its outgoing link, if this link exists. Furthermore, the QNI MUST add the PLR of the ingress link, if this link exists. The composition rule for the <Path PLR> parameter is summation with a clamp on the maximum value (this assumes sufficiently low PLR values such that summation error is not significant, however a more accurate

composition function is specified in clause 8 of [Y.1541]). This quantity, when composed end-to-end, informs the QNR (or QNI in a RESPONSE message) of the minimal packet PLR along the path from QNI to QNR.

The <Path PER> parameter accumulates the packet error rate (PER) of the packet forwarding process associated with each QNE, where the PER is defined to be the PER added by each QNE. Each QNE MUST add the PER of its outgoing link, if this link exists. Furthermore, the QNI MUST add the PER of the ingress link, if this link exists. The composition rule for the <Path PER> parameter is summation with a clamp on the maximum value (this assumes sufficiently low PER values such that summation error is not significant, however a more accurate composition function is specified in clause 8 of [Y.1541]). This quantity, when composed end-to-end, informs the QNR (or QNI in a RESPONSE message) of the minimal packet PER along the path from QNI to QNR.

<Ctot>, <Dtot>, <Csum>, <Dsum>: Error terms C and D represent how the element's implementation of the guaranteed service deviates from the fluid model. These two parameters have an additive composition rule. The error term C is the rate-dependent error term. It represents the delay a datagram in the flow might experience due to the rate parameters of the flow. The error term D is the rate-independent, per-element error term and represents the worst case non-rate-based transit time variation through the service element. If the composition function is applied along the entire path to compute the end-to-end sums of C and D (<Ctot> and <Dtot>) and the resulting values are then provided to the QNR (or QNI in a RESPONSE message). <Csum> and <Dsum> are the sums of the parameters C and D between the last reshaping point and the current reshaping point.

5.3.2.3 <QoS Reserved>

<QoS Reserved> = <Traffic> <QoS Class> <Priority> <S>

Any subset of the QSPEC parameters on the right hand side of the equal sign can be included in the <QoS Reserved> object. These parameters describe the QoS reserved by the QNEs along the data path.

<Traffic>, <QoS Class> and <Priority> are defined above.

<S> = slack term, which is the difference between desired delay and delay obtained by using bandwidth reservation, and which is used to reduce the resource reservation for a flow [RFC2212]. This is an QSPEC-2 parameter.

5.3.2.4 <Minimum QoS>

<Minimum QoS> = <Traffic> <QoS Class> <Priority>

Any subset of the QSPEC parameters on the right hand side of the equal sign can be included in the <Minimum QoS> object.

<Minimum QoS> does not have an equivalent in RSVP. It allows the QNI to define a range of acceptable QoS levels by including both the desired QoS value and the minimum acceptable QoS in the same message. It is a read-only QSPEC object. The desired QoS is included with a <QoS Desired> and/or a <QoS Available> QSPEC object seeded to the desired QoS value. The minimum acceptable QoS value MAY be coded in the <Minimum QoS> QSPEC object. As the message travels towards the QNR, <QoS Available> is updated by QNEs on the path. If its value drops below the value of <Minimum QoS> the reservation fails and is aborted. When this method is employed, the QNR SHOULD signal back to the QNI the value of <QoS Available> attained in the end, because the reservation MAY need to be adapted accordingly.

6. QSPEC Processing & Procedures

The QSPEC is opaque to the QoS NSLP processing, as described in [QoS-SIG]. The QSPEC control information and the QoS description are interpreted by the QNE's RMF and may be modified by the RMF. This section discusses QSPEC processing and how the QNE/RMF interprets QSPEC parameters, stacks QSPECs, determines reservation

success/failure, and signals QSPEC errors and INFO_SPEC notifications. An example of QSPEC processing is given in the final sub-section.

6.1 Interpreting QSPEC Parameters

The QSPEC contains a QOSM ID that identifies which QOSM is being signaled by the QNI. If a QSPEC arrives at a QNE that does not support the QOSM being signaled, it must still interpret the QSPEC content, at least to a basic degree, since QSPEC-1 parameters have been defined as a common language for interoperability of different QOSMs being support in different domains. That is, a QNE must at least interpret all the QSPEC-1 parameters in a QSPEC even if it does not support the corresponding QOSM.

Hence a QNE must either a) strictly interpret a QSPEC parameter, or b) remap, approximate, or otherwise not strictly interpret the QSPEC parameter. Here 'strictly interpret' means that the parameter is implemented by the QNE/RMF according to the commonly accepted procedures as specified by references given for each QSPEC parameter in this document. In the latter case of a remapped QSPEC parameter, the QNE/RMF must raise the remapped parameter R-flag and non-QOSM-hop Q-flag defined in Section 6.3.2, and the remapping must be specified in the QOSM specification. For example, in case a), a <Token Bucket> parameter must be strictly interpreted as a token bucket, and in case b), a <token Bucket> parameter may be remapped to a <Bandwidth> parameter.

In the latter case b), the remapping of the <Token Bucket> to <Bandwidth> must be specified in the QOSM specification document. For example, QOSM X exclusively uses the parameter <Bandwidth>. It must define a mapping of the QSPEC-1 parameter <Token Bucket>. The mapping consists of interpreting the Token Bucket Rate as the <Bandwidth> parameter and disregarding the other Token Bucket parameters. Clearly, some information contained in the <Token Bucket> parameter is lost by this mapping, and the resulting QoS may not be quite what was intended by the QNI. Therefore, QOSM X also specifies that the non-QOSM-hop Q-flag be raised. Thus, a QNE using QOSM X is able to make an informed decision whether to admit a reservation described in terms of <Token Bucket>, and at the same time (by means of the non-QOSM-hop Q-flag) signals to the QNI/QNR that the exact intention of the QNI may not be met.

Other examples of remapping QSPEC-1 parameters are as follows:

- <traffic>: bandwidth remapped to token bucket rate and the other token bucket parameters set to zero or some large value
- <QoS class>: DSTE QoS class to PHB QoS class
- <QoS class>: Y.1541 QoS class remapped to PHB QoS class
- <priority>: admission/RPH = high priority remapping to admission/RPH = normal priority

Remapping between different QSPEC-1 parameter types, e.g., from <QoS Class> to <traffic>, is more complex but is allowed if defined in the QOSM specification document. If a remapping for a QSPEC-1 parameter is not defined in the QOSM specification document, the default is that the QOSM must strictly interpret the QSPEC-1 parameter.

In some cases a QNE may need to reject a reservation because of possible incompatibilities among QSPEC parameters. One example is that some parameters may be illegal (e.g., preemption in the U.S. PSTN). In such a case a QNE must reject a reservation where preemption cannot be accommodated.

6.2 QSPEC Stacking & Tunneling

A QNE at the edge of a local domain may either a) translate the initiator QSPEC into a local QSPEC and stack the local QSPEC on top of the initiator QSPEC in the RESERVE message, or b) tunnel the initiator QSPEC through the local domain and reserve resources by generating a new RESERVE message through the local domain containing the local QSPEC. In either case the initiator QSPEC parameters are interpreted at the local domain edges.

Therefore when reserving resources with a RESERVE message, a local QSPEC MAY be pushed on the stack at the ingress edge of a local QoS domain, in order to describe the requested resources in a domain-specific manner. Here the terms 'ingress' and 'egress' refer to the direction of the RESERVE message rather than the direction of

the flow. Also, the local QSPEC is popped from the stack at the egress edge of the local QoS domain. When a RESPONSE message corresponding to the RESERVE message arrives on its way back at the egress edge, a local QSPEC MUST again be generated, describing the reserved resources in a domain-specific manner. This local QSPEC is popped from the stack at the ingress edge.

A QoS NSLP message can contain a stack of at most two QSPECs. The first on the stack is the initiator QSPEC. This is a QSPEC provided by the QNI, which travels end-to-end, and therefore the stack always has at least depth 1. QSPEC parameters MUST NOT be deleted from or added to the initiator QSPEC. In addition, the stack MAY contain a local QSPEC stacked on top of the initiator QSPEC. A QNE only considers the topmost QSPEC.

QNEs generating a local QSPEC for the purpose of stacking or tunneling have two possible approaches to processing the QSPEC-1 parameters in the initiator QSPEC:

a) The local QSPEC includes all QSPEC-1 parameters in the initiator QSPEC (possibly remapped according to the local QOSM). For example, the initiator QSPEC specifies a token bucket parameter, and it is remapped into the bandwidth parameter in the local QSPEC. The ingress QNE in the local domain does not populate the token bucket parameter in the local QSPEC, rather it populates the bandwidth parameter is the local QSPEC and stacks the local QSPEC on top of the initiator QSPEC. The local QSPEC is interpreted by the QNEs in the local domain, and the egress QNE in the local domain populates token bucket in the initiator QSPEC with just the bandwidth parameter for the token bucket (and not the other token bucket parameters). Note that without QSPEC stacking or tunneling, all QNEs must do this same thing in the local domain, that is, interpret all QSPEC-1 parameters in the initiator QSPEC, which would include remapping the token bucket parameter to the bandwidth parameter.

b) The local QSPEC does not include all QSPEC-1 parameters in the initiator QSPEC, but the egress QNE in the local domain has information configured that allows it to update/process the QSPEC-1 parameters in the initiator QSPEC accordingly. In this case the local QSPEC may carry neither the bandwidth nor token bucket in the above example, if the egress QNE in the local domain has some other means to interpret the token bucket parameter of the initiator QSPEC (e.g., local data base or controller). For example, in a DiffServ domain with a bandwidth broker, the bandwidth broker could inform the egress QNE, or if RSVP is used in the local domain, the information could be obtained from RSVP, or if it is an MPLS domain where LSPs have a particular bandwidth, then the egress QNE knows what is available by counting the reservations that come out of the tunnel. Normally the egress QNE in the local domain interprets the initiator QSPEC parameters,

since doing this in the ingress QNE may require the ingress QNE to inform the egress QNE that it has done this (this is not precluded however).

Note that in the above cases the Q-Flag is set whenever a QOSM is encountered on the path that is different from the Initiator QSPEC, e.g., the Q-Flag is set in both cases a) and b). Also, the R-flag is set for any Initiator QSPEC parameter that is remapped.

QSPEC stacking with a local QSPEC saves interior QNEs from individually interpreting the initiator QSPEC within their local QOSM. Instead, the ingress/egress QNEs do this for them, and in this way consistent processing within a domain is simplified. That is, the equivalent normal behavior is achieved in the local domain as if all QNEs in the domain interpret the initiator QSPEC individually.

6.3 Reservation Success/Failure, QSPEC Error Codes, & INFO_SPEC Notification

A reservation may not be successful for several reasons:

- a reservation may fail because the desired resources are not available. This is a reservation failure condition.

- a reservation may fail because the QSPEC is erroneous, or because of a QNE fault. This is an error condition.

A reservation may be successful even though some parameters could not be interpreted or updated properly:

- a QSPEC parameter cannot be interpreted because it is an unknown QSPEC-2 parameter type. This is a QSPEC parameter not supported condition. The reservation however does not fail. The QNI can still decide whether to keep or tear down the reservation depending on the procedures specified by the QNI's QOSM.

- a QSPEC parameter value is remapped, approximated, or otherwise not strictly interpreted. This is a QSPEC parameter remapped condition. The reservation however does not fail. The QNI can still decide whether to keep or tear down the reservation.

The following sections describe the handling of unsuccessful reservations and reservations where some parameters could not be met in more detail, as follows:

- details on flags used inside the QSPEC to convey information on success or failure of individual parameters. The formats and semantics of all flags are given in Section 7.
- the content of the INFO_SPEC [QoS-SIG], which carries a code indicating the outcome of reservations.

- the generation of a RESPONSE message to the QNI containing both

QSPEC and INFO_SPEC objects.

6.3.1 Reservation Failure & Error E-Flag

The QSPEC parameters each have a 'reservation failure error E-flag' to indicate which (if any) parameters could not be satisfied. When a resource cannot be satisfied for a particular parameter, the QNE detecting the problem raises the E-flag in this parameter. Note that all QSPEC parameters MUST be examined by the RMF and appropriately flagged. Additionally, the E-flag in the corresponding QSPEC object MUST be raised. If the reservation failure problem cannot be located at the parameter level, only the E-flag in the QSPEC object is raised.

When an RMF cannot interpret the QSPEC because the coding is erroneous, it raises corresponding reservation failure E-flags in the QSPEC. Normally all QSPEC parameters MUST be examined by the RMF and the erroneous parameters appropriately flagged. In some cases, however, an error condition may occur and the E-flag of the error-causing QSPEC parameter is raised (if possible), but the processing of further parameters may be aborted.

Note that if the QSPEC and/or any QSPEC parameter is found to be erroneous, then any QSPEC parameters not satisfied are ignored and the E-Flags in the QSPEC object MUST NOT be set for those parameters (unless they are erroneous).

Whether E-flags denote reservation failure or error can be determined by the corresponding error code in the INFO_SPEC in QoS NSLP, as discussed below.

6.3.2 Non-QOSM-Hop Q-Flag & Remapped QSPEC Parameter R-flag

The non-QOSM-hop Q-flag is a flag bit telling the QNR (or QNI in a RESPONSE message) whether or not the initiator QOSM is supported by each QNE in the path between the QNI and QNR. A QNE MUST set the non-QOSM-hop Q-flag parameter if it does not support the relevant initiator QOSM specification. If the QNR finds this bit set, at least one QNE along the data transmission path between the QNI and QNR can not support the specified initiator QOSM. In a local QSPEC, the non-QOSM-hop Q-flag refers to the QoS NSLP peers of the local QOSM domain. When the local QSPEC is popped, the R-Flags of the

corresponding remapped parameters in the initiator QSPEC must be raised. The RESERVE message should continue to be forwarded with the non-QOSM-hop Q-flag set, and the QNI has the option of tearing the reservation.

A QNE detecting that one or more QSPEC parameters have to be remapped, approximated, or otherwise not strictly interpreted MUST set the remapped QSPEC parameter R-flag for each QSPEC parameter that is remapped. The RESERVE message should continue to be forwarded

with the R-flags set, and the QNI has the option of tearing the reservation. This condition might occur, for example, when a QNE's local QOSM is different from the QNI's initiator QOSM, and the local QOSM specifies that some QSPEC parameters are to be remapped. See the example in Appendix A for an illustration of this condition. The R-flag may be interpreted by the QNI, ingress QNE (start of tunnel) in a domain), egress QNE (end of tunnel) in a local domain, or QNR.

When a RESERVE message is tunneled through a local domain, QNEs inside the domain cannot update read-write QSPEC parameters in the initiator QSPEC. The egress QNE in the local domain either a) is configured to have the knowledge to interpret the parameters correctly, or b) cannot accurately interpret the parameters. In the latter case the egress QNE in the local domain MUST set the R-flag for each QSPEC parameter it cannot interpret to tell the QNI (or QNR) that the information contained in the read-write parameter is most likely incorrect (or a lower bound). Note that if possible the edge QNEs in the local domain must interpret the QSPEC-1 parameters populated in the initiator QSPEC and MUST NOT use the R-flag to 'ignore' a QSPEC-1 parameter populated in the initiator QSPEC.

6.3.3 QSPEC Parameter Not Supported N-Flag

When the QOSM ID is not known to a QNE, it MUST interpret at least the QSPEC-1 parameters.

Each QSPEC-2 parameter has an associated 'not supported N-flag'. If the not supported N-flag is set, then at least one QNE along the data transmission path between the QNI and QNR cannot interpret the specified QSPEC-2 parameter. A QNE MUST set the not supported N-flag if it cannot interpret the QSPEC-2 parameter. In that case the message should continue to be forwarded but with the N-flag set, and the QNI has the option of tearing the reservation.

If a QNE in the path does not support a QSPEC-2 parameter, e.g., <Path Latency>, and sets the N-flag, then downstream QNEs that support the parameter SHOULD still update the parameter, even if the N-flag is set. However, the presence of the N-flag will make the cumulative value unreliable, and the QNI/QNR decides whether or not to accept the reservation with the N-flag set.

6.3.4 INFO_SPEC Coding of Reservation Outcome

As prescribed by [QoS-SIG], the RESPONSE message always contains the INFO_SPEC with an appropriate 'error' code. It usually also contains a QSPEC with QSPEC objects, as described in Section 6.3 on QSPEC Procedures. The RESPONSE message MAY omit the QSPEC in case of a successful reservation.

The following guidelines are provided in setting the error codes in the INFO_SPEC, based on the codes provided in Section 5.1.3.6 of

[QoS-SIG]:

- INFO_SPEC error class 0x02 (Success) / 0x01 (Reservation Success):
  This code is set when all QSPEC parameters have been satisfied (possibly with remapping). In this case no E-Flag is set, however the Q-flag, N-flags or R-flags may be set.

- INFO_SPEC error class 0x04 (Transient Failure) / 0x08 (Reservation Failure):
  This code is set when at least one parameter could not be satisfied. E-flags are set for the parameters that could not be satisfied up to the QNE issuing the RESPONSE message. In this case QNEs receiving the RESPONSE message MUST remove the corresponding reservation.

- INFO_SPEC error class 0x03 (Protocol Error) / 0x0c (Malformed QSPEC):
  Some QSPEC parameters had associated errors, E-Flags are set for parameters that had errors, and the RMF rejects the reservation.

- INFO_SPEC error class 0x06 (QoS Model Error):
  QOSM error codes can be defined by QOSM specification documents. A registry is defined in Section 9 IANA Considerations.

6.3.5 QNE Generation of a RESPONSE message

- Successful Reservation Condition

When a RESERVE message arrives at a QNR and no E-Flag is set, the reservation is successful. A RESPONSE message may be generated with INFO_SPEC code 'Reservation Success' as described above and in the QSPEC Procedures described in Section 6.4.

A raised non-QOSM-hop Q-flag in the QSPEC of the RESERVE message indicates that a local QOSM is encountered that differs from the initiator QOSM and that some QSPEC parameters may have been remapped, approximated, or otherwise not strictly interpreted, as indicated by raised R-flags on these QSPEC parameters. The non-QOSM-hop Q-flag and R-flags are sent back in the RESPONSE message and the QNI then makes the final determination as to whether to continue or tear down the reservation that has been established. A QOSM specification may specify the conditions for rejecting a reservation under such conditions. However, in the absence of such procedures, the default condition SHOULD be 'success' if all QSPEC parameters are met and 'reservation failure' if one or more QSPEC parameters are not met.

- Reservation Failure Condition

When a QNE detects that a reservation failure occurs for at least one parameter, the QNE sets the E-Flags for the QSPEC parameters and QSPEC object that failed to be satisfied. According to [QoS-SIG],

the QNE behavior depends on whether it is stateful or not. When a stateful QNE determines the reservation failed, it formulates a RESPONSE message that includes an INFO_SPEC with the 'reservation failure' error code and QSPEC object. The QSPEC in the RESPONSE message includes the failed QSPEC parameters marked with the E-Flag to clearly identify them.

The default action for a stateless QoS NSLP QNE that detects a reservation failure condition is that it MUST continue to forward the RESERVE message to the next stateful QNE, with the E-Flags appropriately set for each QSPEC parameter. The next stateful QNE then formulates the RESPONSE message as described above.

- Malformed QSPEC Error Condition

When a stateful QNE detects that one or more QSPEC parameters are erroneous, the QNE sets the error code 'malformed QSPEC' in the INFO_SPEC. In this case the QSPEC object with the E-Flags

appropriately set for the erroneous parameters is returned within the INFO_SPEC object. The QSPEC object can be truncated or fully included within the INFO_SPEC.

According to [QoS-SIG], the QNE behavior depends on whether it is stateful or not. When a stateful QNE determines a malformed QSPEC error condition, it formulates a RESPONSE message that includes an INFO_SPEC with the 'malformed QSPEC' error code and QSPEC object. The QSPEC in the RESPONSE message includes, if possible, only the erroneous QSPEC parameters and no others. The erroneous QSPEC parameter(s) are marked with the E-Flag to clearly identify them. If QSPEC parameters are returned in the INFO-SPEC that are not marked with the E-flag, then any values of these parameters are irrelevant and MUST be ignored by the QNI.

The default action for a stateless QoS NSLP QNE that detects a Malformed QSPEC error condition is that it MUST continue to forward the RESERVE message to the next stateful QNE, with the E-Flags appropriately set for each QSPEC parameter. The next stateful QNE will then act as described in [QoS-SIG].

A 'malformed QSPEC' error code takes precedence over the 'reservation failure' error code, and therefore the case of reservation failure and QSPEC/RMF error conditions are disjoint and the same E-Flag can be used in both cases without ambiguity.

6.3.6 Domains Supporting a Different Local QOSM than the QNI

A domain supporting a different local QOSM than the QNI domain inspects all QSPEC-1 parameters and consults its local QOSM as to how to interpret these parameters and decides whether it can accommodate the flow. This analysis can have these various outcomes:

a) RMF determines that it can accommodate the flow with the QoS specified by the QNI,
b) RMF determines that some initiator QSPEC parameters cannot be satisfied with the available resources, and marks the appropriate error flags (see Section 6.3.1), but does not reject the reservation, or
c) RMF determines that some initiator QSPEC parameters cannot be satisfied with the available resources, marks the appropriate error flags (see Section 6.3.1), and also rejects the reservation. The QNE also in any event sets the non-QOSM-hop Q-flag, as

described in Section 6.3.2.

6.3.7 Special Cases of QSPEC Stacking

When an unsuccessful reservation problem occurs inside a local domain where QSPEC stacking is used, only the topmost (local) QSPEC is affected (e.g. E-flags are raised, etc.). The initiator QSPEC at the bottom is untouched. When the message (RESPONSE in case of stateful QNEs, RESERVE in case of stateless QNEs) however reaches the edge of the stacking domain, the local QSPEC is popped, and its content, including flags, is translated into the initiator QSPEC.

6.4 QSPEC Procedures

While the QSPEC template aims to put minimal restrictions on usage of QSPEC objects in <QoS Description>, interoperability between QNEs and between QOSMs must be ensured. We therefore give below an exhaustive list of QSPEC object combinations for the message sequences described in QoS NSLP [QoS-SIG]. A specific QOSM may prescribe that only a subset of the procedures listed below may be used.

Note that QoS NSLP does not mandate the usage of a RESPONSE message. In fact, a RESPONSE message will only be generated if the QNI includes an RII (Request Identification Information) in the RESERVE message. Some of the QSPEC procedures below, however, are only meaningful when a RESPONSE message is possible. The QNI SHOULD in these cases include an RII.

6.4.1 Sender-Initiated Reservations

Here the QNI issues a RESERVE message, which may be replied to by a RESPONSE message. The following 3 cases for QSPEC object usage exist:

ID	RESERVE	RESPONSE
1	QoS Desired	QoS Reserved
2	QoS Desired, QoS Avail.	QoS Reserved, QoS Avail.
3	QoS Desired, QoS Avail., Min. QoS	QoS Reserved, QoS Avail.

Case 1:

If only QoS Desired is included in the RESERVE message, the implicit

assumption is that exactly these resources must be reserved. If this is not possible the reservation fails. The parameters in QoS Reserved are copied from the parameters in QoS Desired. If the reservation is successful, the RESPONSE message can be omitted in this case. If a RESPONSE message was requested by a QNE on the path, the QSPEC in the RESPONSE message can be omitted.

Case 2:

When QoS Available is included in the RESERVE message also, some parameters will appear only in QoS Available and not in QoS Desired. It is assumed that the value of these parameters is collected for informational purposes only (e.g. path latency).

However, some parameters in QoS Available can be the same as in QoS Desired. For these parameters the implicit message is that the QNI would be satisfied by a reservation with lower parameter values than specified in QoS Desired. For these parameters, the QNI seeds the parameter values in QoS Available to those in QoS Desired (except for cumulative parameters such as <path latency>).

Each QNE remaps or approximately interprets the parameters in QoS Available according to its current capabilities. Reservations in each QNE are hence based on current parameter values in QoS Available (and additionally those parameters that only appear in QoS Desired). The drawback of this approach is that, if the resulting resource reservation becomes gradually smaller towards the QNR, QNEs close to the QNI have an oversized reservation, possibly resulting in unnecessary costs for the user. Of course, in the RESPONSE the QNI learns what the actual reservation is (from the QoS RESERVED object) and can immediately issue a properly sized refreshing RESERVE. The advantage of the approach is that the reservation is performed in half-a-roundtrip time.

The QSPEC parameter IDs and values included in the QoS Reserved object in the RESPONSE message MUST be the same as those in the QoS Desired object in the RESERVE message. For those QSPEC parameters that were also included in the QoS Available object in the RESERVE message, their value is copied into the QoS Desired object. For the other QSPEC parameters, the value is copied from the QoS Desired object (the reservation would fail if the corresponding QoS could not be reserved).

All parameters in the QoS Available object in the RESPONSE message are copied with their values from the QoS Available object in the RESERVE message (irrespective of whether they have also been copied into the QoS Desired object). Note that the parameters in the QoS Available object are read-write in the RESERVE message, whereas they

are read-only in the RESPONSE message.

In this case, the QNI SHOULD request a RESPONSE message since it will otherwise not learn what QoS is available.

Case 3:

This case is handled as case 2, except that the reservation fails when QoS Available becomes less than Minimum QoS for one parameter. If a parameter appears in the QoS Available object but not in the Minimum QoS object it is assumed that there is no minimum value for this parameter.

Regarding QSPEC Control Information, the default rule is that all QSPEC parameters that have been included in the RESERVE message by the QNI are also included in the RESPONSE message by the QNR with the value they had when arriving at the QNR. When traveling in the RESPONSE message, all QSPEC Control Information parameters are read-only. Note that a QOSM specification may define its own QOSM-specific Control Information parameters and processing rules. Also in this case, the QNI SHOULD request a RESPONSE message since it will otherwise not learn what QoS is available.

6.4.2 Receiver-Initiated Reservations

Here the QNR issues a QUERY message which is replied to by the QNI with a RESERVE message if the reservation was successful. The QNR in turn sends a RESPONSE message to the QNI. The following 3 cases for QSPEC object usage exist:

ID	QUERY	RESERVE	RESPONSE
1	QoS Des.	QoS Des.	QoS Res.
2	QoS Des.,Min. QoS	QoS Des.,QoS Avl.,(Min QoS)	QoS Res.,QoS Avl.
3	Qos Des.,QoS Avl.	QoS Des.,QoS Avl.	QoS Res.

Cases 1 and 2:

The idea is that the sender (QNR in this scenario) needs to inform the receiver (QNI in this scenario) about the QoS it desires. To this end the sender sends a QUERY message to the receiver including a QoS Desired QSPEC object. If the QoS is negotiable it additionally includes a (possibly zero) Minimum QoS object, as in Case 2.

The RESERVE message includes the QoS Available object if the sender signaled that QoS is negotiable (i.e. it included the Minimum QoS object). If the Minimum QoS object received from the sender is

included in the QUERY message, the QNR also includes the Minimum QoS object in the RESERVE message.

For a successful reservation, the RESPONSE message in case 1 is

optional (as is the QSPEC inside). In case 2 however, the RESPONSE message is necessary in order for the QNI to learn about the QoS available.

Case 4:

This is the 'RSVP-style' scenario. The sender (QNR in this scenario) issues a QUERY message with a QoS Desired object informing the receiver (QNI in this scenario) about the QoS it desires as above. It also includes a QoS Available object to collect path properties. Note that here path properties are collected with the QUERY message, whereas in the previous case 2 path properties were collected in the RESERVE message.

Some parameters in the QoS Available object may the same as in the QoS Desired object. For these parameters the implicit message is that the sender would be satisfied by a reservation with lower parameter values than specified in QoS Desired.

It is possible for the QoS Available object to contain parameters that do not appear in the QoS Desired object. It is assumed that the value of these parameters is collected for informational purposes only (e.g. path latency). Parameter values in the QoS Available object are seeded according to the sender's capabilities. Each QNE remaps or approximately interprets the parameter values according to its current capabilities.

The receiver (QNI in this scenario) signals the QoS Desired object as follows: For those parameters that appear in both the QoS Available object and QoS Desired object in the QUERY message, it takes the (possibly remapped) QSPEC parameter values from the QoS Available object. For those parameters that only appear in the QoS Desired object, it adopts the parameter values from the QoS Desired object.

The parameters in the QoS Available QSPEC object in the RESERVE message are copied with their values from the QoS Available QSPEC object in the QUERY message. Note that the parameters in the QoS Available object are read-write in the QUERY message, whereas they

are read-only in the RESERVE message.

The advantage of this model compared to the sender-initiated reservation is that the situation of over-reservation in QNEs close to the QNI as described above does not occur. On the other hand, the QUERY message may find, for example, a particular bandwidth is not available. When the actual reservation is performed, however, the desired bandwidth may meanwhile have become free. That is, the 'RSVP style' may result in a smaller reservation than necessary.

Regarding QSPEC Control Information in receiver-initiated reservations, the sender includes all QSPEC Control Information it

cares about in the QUERY message. Read-write parameters are updated by QNEs as the QUERY message travels towards the receiver. The receiver includes all QSPEC Control Information parameters arriving in the QUERY message also in the RESERVE message, as read-only parameters with the value they had when arriving at the receiver. Again, QOSM-specific Control Information parameters and procedures may be defined in QOSM specification documents.

Also in this scenario, the QNI SHOULD request a RESPONSE message since it will otherwise not learn what QoS is available.

6.4.3 Resource Queries

Here the QNI issues a QUERY message in order to investigate what resources are currently available. The QNR replies with a RESPONSE message.

ID	QUERY	RESPONSE
1	QoS Available	QoS Available

Note that the QoS Available object when traveling in the QUERY message is read-write, whereas in the RESPONSE message it is read-only.

6.4.4 Bidirectional Reservations

On a QSPEC level, bidirectional reservations are no different from uni-directional reservations, since QSPECs for different directions never travel in the same message.

6.4.5 Preemption

A flow can be preempted by a QNE based on the values of the QSPEC Priority parameter (see Section 7.2.4). In this case the reservation state for this flow is torn down in this QNE, and the QNE sends a NOTIFY message to the QNI, as described in [QoS-SIG]. No QSPEC is carried in the NOTIFY message. The NOTIFY message carries only the Session ID and a INFO_SPEC with the error code as described in [QoS-SIG]. The QNI would normally tear down the preempted reservation by sending a RESERVE message with the TEAR flag set using the SII of the preempted reservation. However, the QNI can follow other procedures as specified in its QOSM specification document.

6.5 QSPEC Extensibility

This document defines both QSPEC-1 and QSPEC-2 parameters. The set of QSPEC-1 parameters defined herein is at this point in time considered complete. The QSPEC-2 parameters in this document correspond to some of the QSPEC-2 parameters considered in QOSMs currently being defined.

Additional QSPEC-1 parameters may be defined in the future. However, since this requires an update of all QNEs, this should be considered carefully. The definition of new QSPEC-1 parameter requires standards action and an update of this document. Such an update also needs a new QSPEC version number. Furthermore, all QOSM definitions must be updated to include how the new QSPEC-1 parameter is to be interpreted in the respective QOSM.

Additional QSPEC-2 parameters MAY need to be defined in the future and are defined in separate informational documents specific to a given QOSM. For example, QSPEC-2 parameters are defined in [RMD-QOSM] and [Y.1541-QOSM].

Guidelines on the technical criteria to be followed in evaluating requests for new codepoint assignments for QSPEC objects and QSPEC parameters are given in Section 9 (IANA Considerations).

Guidelines on the technical criteria to be followed in evaluating requests for new codepoint assignments beyond QSPEC objects and QSPEC parameters for the NSIS protocol suite are given in a separate NSIS extensibility document [NSIS-EXTENSIBILITY].

7. QSPEC Functional Specification

This section defines the encodings of the QSPEC parameters and QSPEC control information defined in Section 5. We first give the general QSPEC formats and then the formats of the QSPEC objects and parameters.

Note that all QoS description parameters can be either read-write or read-only, depending on which object and which message they appear in. All parameters in the QoS Desired object, QoS Reserved object, and Minimum QoS object are read-only for all messages. All parameters in the QoS Available object are normally read-write parameters. However, as discussed in Section 5.2.2, the parameters in the QoS Available object are read-write when the QoS Available object appears for the first time e.g. in the RESERVE message or QUERY message from QNI to QNR. However, on its way back, all parameters in the <QoS Available> object are read-only, e.g., in the RESPONSE message or RESERVE message from QNR to QNI. For QSPEC control information parameters, the property of being read-write or read-only is parameter specific. Note that the only control information parameter specified in this document is the <excess treatment> parameter, which is a read-only parameter.

Network byte order ('big-endian') for all 16- and 32-bit integers, as well as 32-bit floating point numbers, are as specified in [RFC1832, IEEE754, NETWORK-BYTE-ORDER].

7.1 General QSPEC Formats

The format of the QSPEC closely follows that used in GIST [GIST] and QoS NSLP [QoS-SIG]. Every object (and parameter) has the following general format:

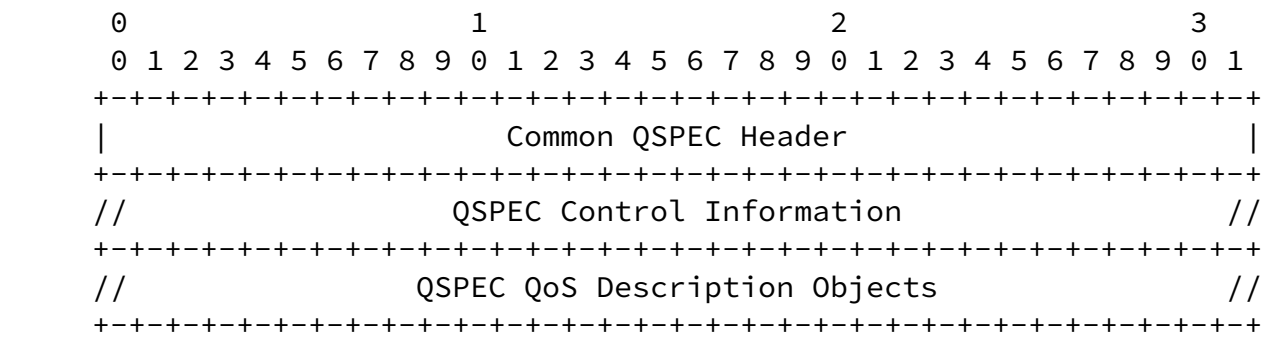
- The overall format is Type-Length-Value (in that order).
- Some parts of the type field are set aside for control flags.
- Length has the units of 32-bit words, and measures the length of Value. If there is no Value, Length=0. The Object length excludes the header.
- Value is a whole number of 32-bit words. If there is any padding required, the length and location MUST be defined by the

object-specific format information; objects that contain variable length types may need to include additional length subfields to do so.

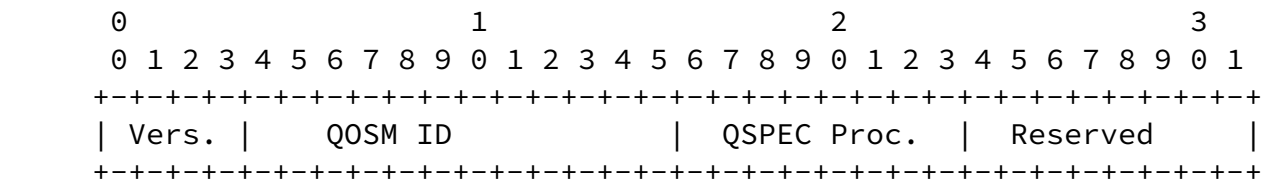
- Any part of the object used for padding or defined as reserved("r") MUST be set to 0 on transmission and MUST be ignored on reception.

- Empty QSPECs and empty QSPEC Objects MUST NOT be used.

- Duplicate objects, duplicate parameters, and/or multiple occurrences of a parameter MUST NOT be used.



The Common QSPEC Header is a fixed 4-byte long object containing the QOSM ID and an identifier for the QSPEC Procedure (see Section 6.4):



Note that a length field is not necessary since the overall length of the QSPEC is contained in the higher level QoS NSLP data object.

Vers.: Identifies the QSPEC version number. It is assigned by IANA.

QOSM ID: Identifies the particular QOSM being used by the QNI. It is assigned by IANA.

QSPEC Proc.: Is composed of two times 4 bits. The first set of bits identifies the Message Sequence, the second set identifies the QSPEC Object Combination used for this particular message sequence:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|Mes.Sq |Obj.Cmb|
+-+-+-+-+-+-+-+-+
```

The Message Sequence field can attain the following values:

0: Sender-Initiated Reservations
1: Receiver-Initiated Reservations
2: Resource Queries

The Object Combination field can take the values between 1 and 3 indicated in the tables in Section 6.4:
Message Sequence: 0
Object Combination: 1, 2, 3
Semantic: see table in Section 6.4.1
Message Sequence: 1
Object Combination: 1, 2, 3
Semantic: see table in Section 6.4.2
Message Sequence: 2
Object Combination: 1, 2, 3
Semantic: see table in Section 6.4.3

The QSPEC Control Information is a variable length object containing one or more parameters. The QSPEC Objects field is a collection of QSPEC objects (QoS Desired, QoS Available, etc.), which share a common format and each contain several parameters.

Both the QSPEC Control Information object and the QSPEC QoS objects share a common header format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|Q|r|r|       Object Type     |r|r|r|r|         Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

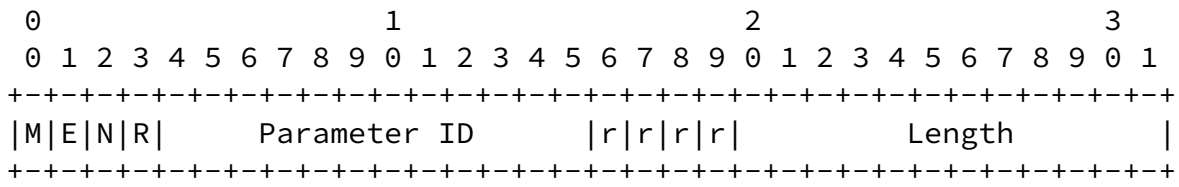
E Flag: Set if an error occurs on object level
Q Flag: NON QOSM Hop flag: This field is set to 1 if a QOSM different from the initiator QOSM is encountered by the QNE.
Object Type = 0: control information (read-only/read-write status is

parameter specific)
= 1: QoS Desired (parameters are all read-only)
= 2: QoS Available (parameters are either all read-write
rr all read-only; see Section 5.2.2)
= 3: QoS Reserved (parameters are all read-only)
= 4: Minimum QoS (parameters are all read-only)

Note that parameters contained in QoS Description objects are all read-write or all read-only, as specified above. In the Control Information object, read-only or read-write is parameter specific.

The r bits are reserved.

Each QSPEC-1 or QSPEC-2 parameter within an object is similarly encoded in TLV format using a similar parameter header:

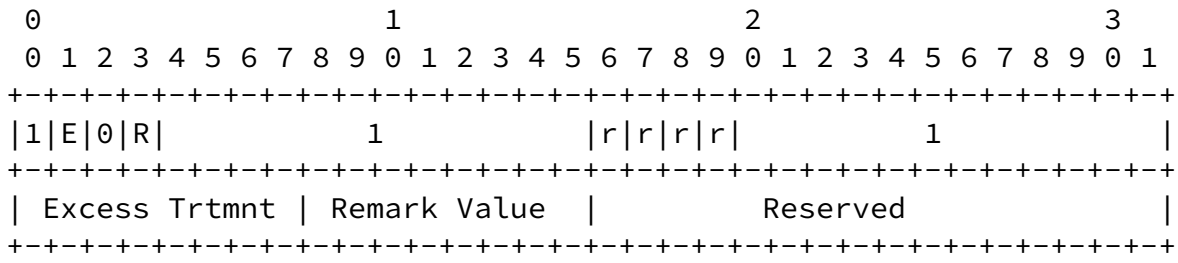


M Flag: When set indicates the subsequent parameter is a QSPEC-1 parameter and MUST be interpreted. Otherwise the parameter is QSPEC-2 and can be ignored if not understood.
E Flag: When set indicates either a) a reservation failure where the QSPEC parameter is not met, or b) an error occurred when this parameter was being interpreted (see Section 6.3.1).
N Flag: Not-supported QSPEC parameter flag (see Section 6.3.3). For QSPEC-1 parameters the value of this flag is always zero.
R Flag: Remapped, approximated, or otherwise not strictly interpreted QSPEC parameter flag (see Section 6.3.2)
Parameter ID: Assigned to each parameter (see below)

Parameters are usually coded individually, for example, the <Excess Treatment> parameter (Section 7.2.1). However, it is also possible to combine several sub-parameters into one parameter field, which is called 'container coding'. This coding is useful if either a) the sub-parameters always occur together, as for example the several sub-parameters that jointly make up the token bucket, or b) in order to make coding more efficient when the length of each sub-parameter value is much less than a 32-bit word (as for example described in [RMD-QOSM]) and to avoid header overload. When a container is defined, the Parameter ID and the M, E, N, and R flags refer to the container. Examples of container parameters are <Traffic>, <QoS Class>, <Priority>, and <Token Bucket>, as specified below, and the PHR container parameter specified in [RMD-QOSM].

7.2 QSPEC-1 Parameter Coding

7.2.1 <Excess Treatment> Parameter



Excess Treatment: Indicates how the QNE SHOULD process out-of-profile traffic, that is, traffic not covered by the <Traffic> parameter. The excess treatment parameter is set by the QNI. It is a read-only parameter. Allowed values are as follows:

0: drop
1: shape
2: remark
3: no metering or policing is permitted

The default excess treatment in case that none is specified is that there are no guarantees to excess traffic, i.e. a QNE can do whatever it finds suitable.

When excess treatment is set to 'drop', all marked traffic MUST be dropped by the QNE/RMF.

When excess treatment is set to 'shape', it is expected that the QoS Desired object carries a token bucket parameter. Excess traffic is to be shaped to this token bucket. When the shaping causes unbounded queue growth at the shaper traffic can be dropped.

When excess treatment is set to 'remark', the excess treatment parameter MUST carry the remark value, and the remark values and procedures MUST be specified in the QOSM specification document. For example, packets may be remarked to drop remarked to pertain to a particular QoS class". In the latter case, remarking relates to a DiffServ-type model, where packets arrive marked as belonging to a certain QoS class, and when they are identified as excess, they should then be remarked to a different QoS Class.

If 'no metering or policing is permitted' is signaled, the QNE should accept the excess treatment parameter set by the sender with special care so that excess traffic should not cause a problem. To request the Null Meter [RFC3290] is especially strong, and should be used

with caution.

A NULL metering application [RFC2997] would not include the traffic profile, and conceptually it should be possible to support this with

the QSPEC. A QSPEC without a traffic profile is not excluded by the current specification. However, note that the traffic profile is important even in those cases when the excess treatment is not specified, e.g., in negotiating bandwidth for the best effort aggregate. However, a "NULL Service QOSM" would need to be specified where the desired QNE Behavior and the corresponding QSPEC format are described.

As an example behavior for a NULL metering, in the properly configured DiffServ router, the resources are shared between the aggregates by the scheduling disciplines. Thus, if the incoming rate increases, it will influence the state of a queue within that aggregate, while all the other aggregates will be provided sufficient bandwidth resources. NULL metering is useful for best effort and signaling data, where there is no need to meter and police this data as it will be policed implicitly by the allocated bandwidth and, possibly, active queue management mechanism.

7.2.2 <Traffic> Parameter

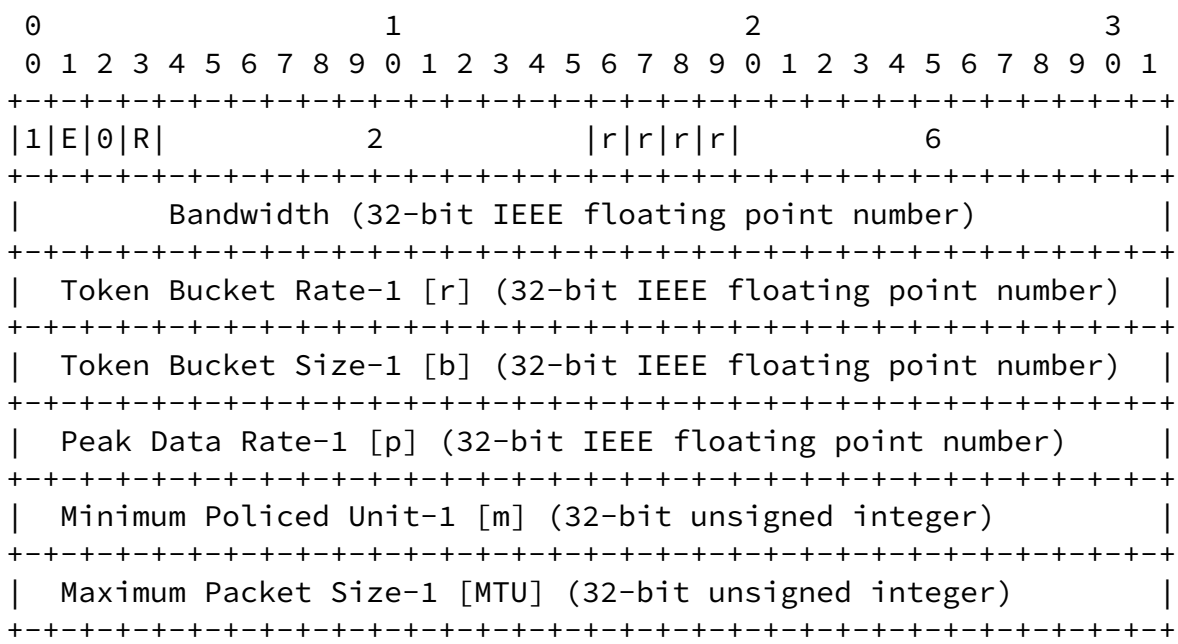
<Traffic> = [<Bandwidth>] [<Token Bucket-1>]

<Bandwidth> = link bandwidth needed by flow [RFC2212, RFC2215]

<Token Bucket-1> = <r> <b> <p> <m> <MTU> [RFC2210]

The above notation means that either <Bandwidth> or <Token Bucket-1> sub-parameters can be populated in the <Traffic> parameter and that one and only one of them MUST be present. Note that an QSPEC-2 second token bucket QSPEC parameter <Token Bucket-2> is specified below in Section 7.3.1. The references in the following sections point to the normative procedures for processing the <Bandwidth> and <Token Bucket> sub-parameters.

The coding for the <Traffic> parameter is as follows:



Normally only one of these sub-parameters is populated in the <Traffic> parameter. If more than one sub-parameter is populated, the QOSM specification document MUST give procedures for processing multiple sub-parameters. The references in the following sections point to the normative procedures for processing the <Bandwidth> and <Token Bucket-1> sub-parameters.

7.2.2.1 <Bandwidth> Sub-Parameter [RFC2212, RFC2215]

The <Bandwidth> parameter MUST be nonnegative and is measured in bytes per second and has the same range and suggested representation as the bucket and peak rates of the <Token Bucket>. <Bandwidth> is represented using single-precision IEEE floating point. The representation MUST be able to express values ranging from 1 byte per second to 40 terabytes per second. For values of this parameter only valid non-negative floating point numbers are allowed. Negative numbers (including "negative zero"), infinities, and NAN's are not allowed.

A QNE MAY export a local value of zero for this parameter. A network element or application receiving a composed value of zero for this parameter MUST assume that the actual bandwidth available is unknown.

7.2.2.2 <Token Bucket-1> Sub-Parameters [RFC2215]

The <Token Bucket> parameters are represented by three floating point numbers in single-precision IEEE floating point format followed by two 32-bit integers in network byte order. The first floating point value is the rate (r), the second floating point value is the bucket

unsigned integer is the maximum datagram size (MTU).

Note that the two sets of <Token Bucket> parameters can be distinguished, as could be needed for example to support DiffServ applications.

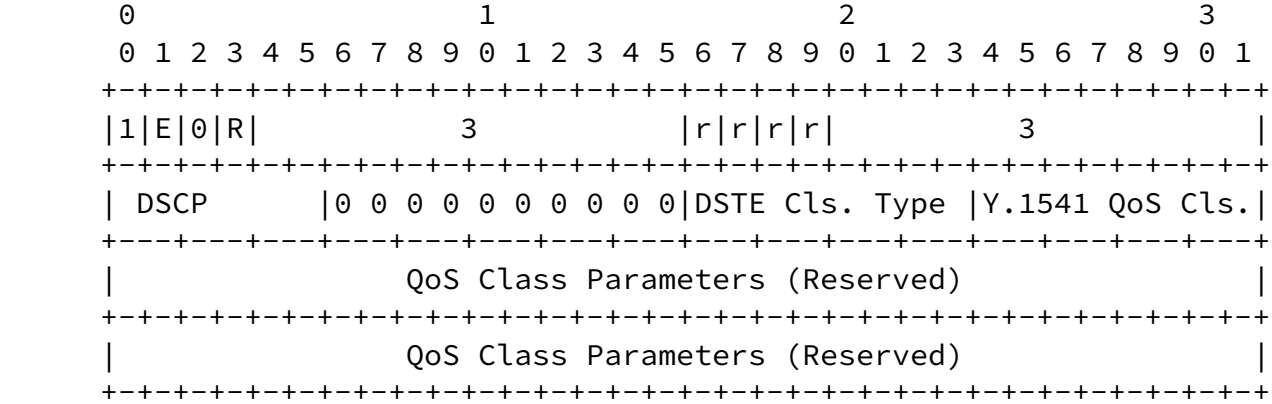
When r, b, and p terms are represented as IEEE floating point values, the sign bit MUST be zero (all values MUST be non-negative). Exponents less than 127 (i.e., 0) are prohibited. Exponents greater than 162 (i.e., positive 35) are discouraged, except for specifying a peak rate of infinity. Infinity is represented with an exponent of all ones (255) and a sign bit and mantissa of all zeroes.

7.2.3 <QoS Class> Parameter

<QoS Class> = [<PHB Class>] [<DSTE Class Type>] [<Y.1541 QoS Class>]

The above notation means that either <PHB Class>, <DSTE Class Type>, or <Y.1541 QoS Class> sub-parameters MAY be populated in the <QoS Class> parameter. Normally only one of these sub-parameters is populated in <QoS Class>. If more than one sub-parameter is populated, the QOSM specification document MUST give procedures for processing multiple sub-parameters. The references in the following sections point to the normative procedures for processing the <PHB Class>, <DSTE Class Type>, and <Y.1541 QoS Class> sub-parameters.

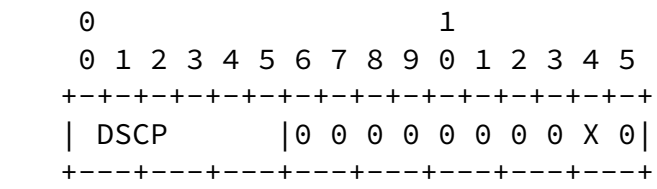
The coding for the <QoS Class> parameter is as follows:



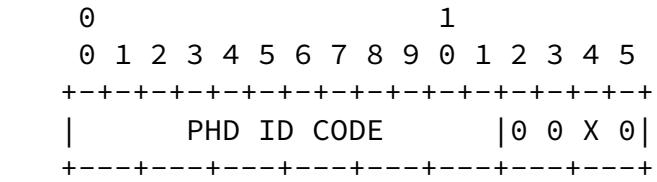
7.2.3.1 <PHB Class> Sub-Parameter [RFC3140]

As prescribed in RFC 3140, the encoding for a single PHB is the recommended DSCP value for that PHB, left-justified in the 16 bit field, with bits 6 through 15 set to zero.

The encoding for a set of PHBs is the numerically smallest of the set of encodings for the various PHBs in the set, with bit 14 set to 1. (Thus for the AF1x PHBs, the encoding is that of the AF11 PHB, with bit 14 set to 1.)



PHBs not defined by standards action, i.e., experimental or local use PHBs as allowed by [RFC2474]. In this case an arbitrary 12 bit PHB identification code, assigned by the IANA, is placed left-justified in the 16 bit field. Bit 15 is set to 1, and bit 14 is zero for a single PHB or 1 for a set of PHBs. Bits 12 and 13 are zero.



Bits 12 and 13 are reserved either for expansion of the PHB identification code, or for other use, at some point in the future.

In both cases, when a single PHBID is used to identify a set of PHBs (i.e., bit 14 is set to 1), that set of PHBs MUST constitute a PHB Scheduling Class (i.e., use of PHBs from the set MUST NOT cause intra-microflow traffic reordering when different PHBs from the set are applied to traffic in the same microflow). The set of AF1x PHBs [RFC2597] is an example of a PHB Scheduling Class. Sets of PHBs that do not constitute a PHB Scheduling Class can be identified by using more than one PHBID.

The registries needed to use RFC 3140 already exist, see [DSCP-REGISTRY, PHBID-CODES-REGISTRY]. Hence, no new registry needs to be created for this purpose.

7.2.3.2 <DSTE Class Type> Sub-Parameter [RFC4124]

DSTE Class Type: Indicates the DSTE class type. Values currently allowed are 0, 1, 2, 3, 4, 5, 6, 7. A value of 255 (all 1's) means that the <DSTE Class Type> parameter is not used.

7.2.3.3 <Y.1541 QoS Class> Sub-Parameter [Y.1541]

Y.1541 QoS Class: Indicates the Y.1541 QoS Class. Values currently allowed are 0, 1, 2, 3, 4, 5, 6, 7. A value of 255 (all 1's) means that the <Y.1541 QoS Class> parameter is not used.

Class 0:
Mean delay <= 100 ms, delay variation <= 50 ms, loss ratio <= 10^-3. Real-time, highly interactive applications, sensitive to jitter. Application examples include VoIP, Video Teleconference.

Class 1:
Mean delay <= 400 ms, delay variation <= 50 ms, loss ratio <= 10^-3. Real-time, interactive applications, sensitive to jitter. Application examples include VoIP, Video Teleconference.

Class 2:
Mean delay <= 100 ms, delay variation unspecified, loss ratio <= 10^-3. Highly interactive transaction data. Application examples include signaling.

Class 3:
Mean delay <= 400 ms, delay variation unspecified, loss ratio <= 10^-3. Interactive transaction data. Application examples include signaling.

Class 4:
Mean delay <= 1 sec, delay variation unspecified, loss ratio <= 10^-3. Low Loss Only applications. Application examples include short transactions, bulk data, video streaming.

Class 5:
Mean delay unspecified, delay variation unspecified, loss ratio unspecified. Unspecified applications. Application examples include traditional applications of default IP networks.

Class 6:
Mean delay <= 100 ms, delay variation <= 50 ms, loss ratio <= 10^-5.
Applications that are highly sensitive to loss, such as television transport, high-capacity TCP transfers, and TDM circuit emulation.

Class 7:
Mean delay <= 400 ms, delay variation <= 50 ms, loss ratio <= 10^-5.
Applications that are highly sensitive to loss, such as television transport, high-capacity TCP transfers, and TDM circuit emulation.

7.2.4 <Priority> Parameter

```
<Priority> = [<Preemption Priority>] [<Defending Priority>]
             [<Admission Priority>] [<RPH Priority>]
```

The above notation means that either <Preemption Priority>, <Defending Priority>, <Admission Priority>, and/or <RPH Priority> sub-parameters MAY be populated in the <Priority> parameter. Any or all of these sub-parameters may be populated in the <Priority> parameter. The references in the following sections point to the normative procedures for processing the <Preemption Priority>, <Defending Priority>, <Admission Priority>, and <RPH Priority> sub-parameters.

The following cases are permissible (procedures specified in references):

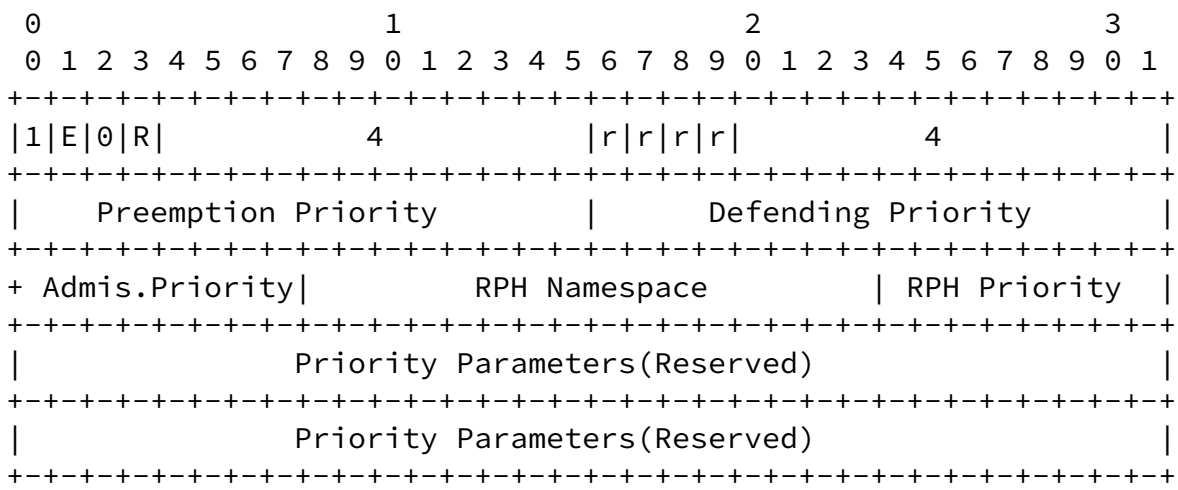
1 parameter: <Admission Priority> [Y.1571]
2 parameters: <Admission Priority>, <RPH Priority> [RFC4412]
2 parameters: <Preemption Priority>, <Defending Priority> [RFC3181]
3 parameters: <Preemption Priority>, <Defending Priority>, <Admission Priority> [3GPP-1, 3GPP-2, 3GPP-3]
4 parameers: <Preemption Priority>, <Defending Priority>, <Admission Priority>, <RPH Priority> [3GPP-1, 3GPP-2, 3GPP-3]

It is permissible to have <Admission Priority> without <RPH Priority>, but not permissible to have <RPH Priority> without <Admission Priority> (alternatively <RPH Priority> is ignored in instances without <Admission Priority>).

eMLPP-like functionality (as defined in [3GPP-1, 3GPP-2]) specifies use of <Admission Priority> corresponding to the 'queuing allowed'

part of eMLPP as well as <Preemption/Defending Priority> corresponding to the 'preemption capable' and 'may be preempted' parts of eMLPP.

The coding for the <Priority> parameter is as follows:



7.2.4.1 <Preemption Priority> & <Defending Priority> Sub-Parameters [RFC3181]

Preemption Priority: The priority of the new flow compared with the defending priority of previously admitted flows. Higher values represent higher priority.

Defending Priority: Once a flow is admitted, the preemption priority becomes irrelevant. Instead, its defending priority is used to compare with the preemption priority of new flows.

As specified in [RFC3181], <Preemption Priority> and <Defending Priority> are 16-bit integer values and both MUST be populated if the parameter is used.

7.2.4.2 <Admission Priority> Sub-Parameter [Y.1571]

High priority flows, normal priority flows, and best-effort priority flows can have access to resources depending on their admission priority value, as described in [Y.1571], as follows:

Admission Priority:

0 - best-effort priority flow
1 - normal priority flow
2 - high priority flow
255 - not used

A reservation without an <Admission Priority> sub-parameter (i.e., set to 255) MUST be treated as a reservation with an <Admission Priority> = 1.

7.2.4.3 <RPH Priority> Sub-Parameter [RFC4412]

[RFC4412] defines a resource priority header (RPH) with parameters "RPH Namespace" and "RPH Priority" combination, and if populated is applicable only to flows with high admission priority, as follows:

RPH Namespace:

0 - dsn
1 - drsn
2 - q735
3 - ets
4 - wps
255 - not used

RPH Priority:
Each namespace has a finite list of relative priority-values. Each is listed here in the order of lowest priority to highest priority (note that dsn and drsn priority values are TBD):

4 - q735.4
3 - q735.3
2 - q735.2
1 - q735.1
0 - q735.0

4 - ets.4
3 - ets.3
2 - ets.2
1 - ets.1
0 - ets.0

4 - wps.4

3 - wps.3

2 - wps.2
1 - wps.1
0 - wps.0

Note that the <Admission Priority> parameter MAY be used in combination with the <RPH Priority> parameter, which depends on the supported QOSM. Furthermore, if more then one RPH namespace is supported by a QOSM, then the QOSM MUST specify how the mapping between the priorities belonging to the different RPH namespaces are mapped to each other.

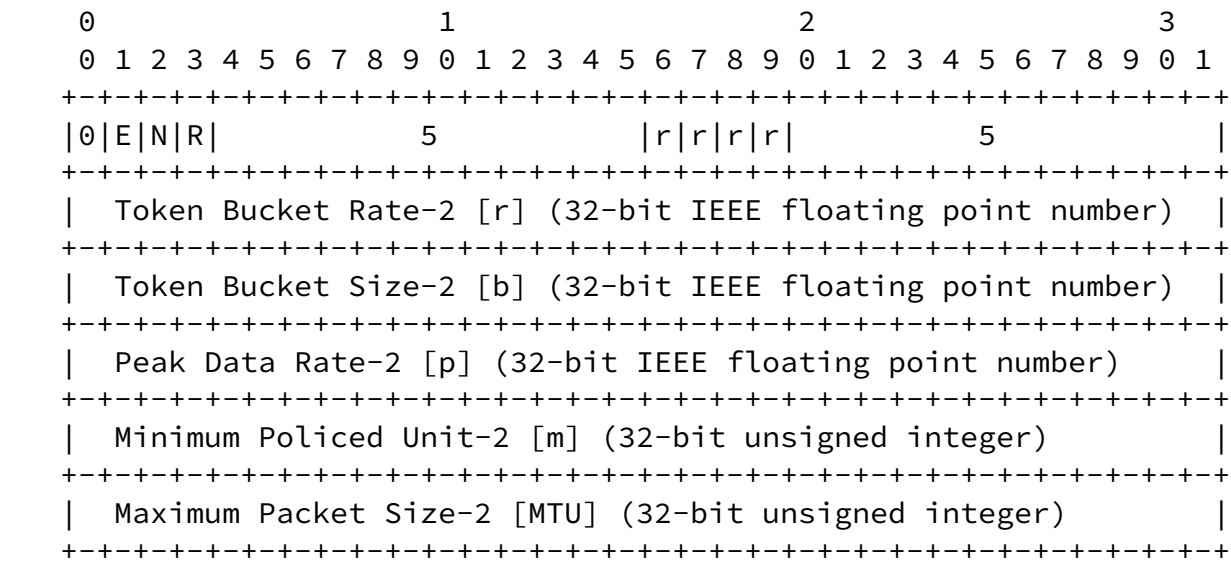
Note also that additional work is needed to communicate these flow priority values to bearer-level network elements [VERTICAL-INTERFACE].

7.3 QSPEC-2 Parameter Coding

7.3.1 <Token Bucket-2> Parameter [RFC2215]

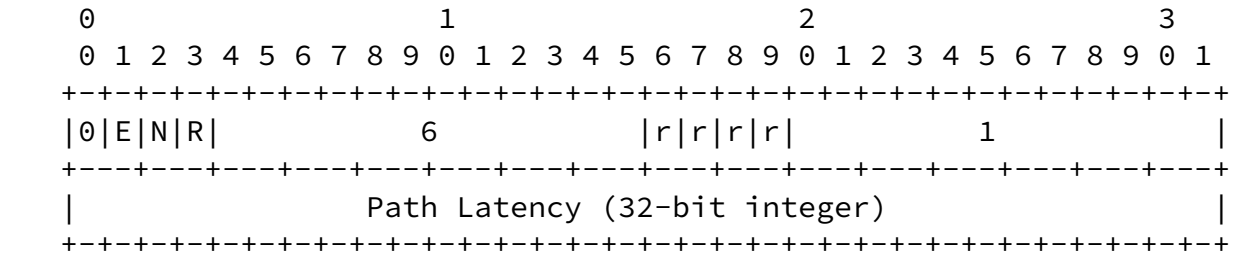
A second, QSPEC-2 <Token Bucket-2> parameter is specified, as could be needed for example to support DiffServ applications [xxxx].

Parameter Values:



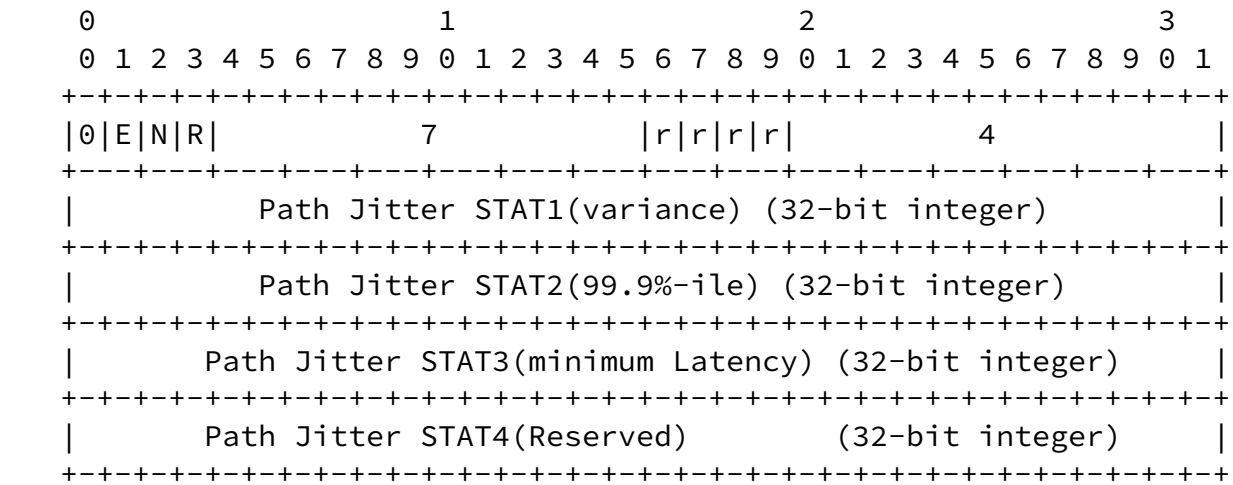
When r, b, and p terms are represented as IEEE floating point values, the sign bit MUST be zero (all values MUST be non-negative). Exponents less than 127 (i.e., 0) are prohibited. Exponents greater than 162 (i.e., positive 35) are discouraged, except for specifying a peak rate of infinity. Infinity is represented with an exponent of all ones (255) and a sign bit and mantissa of all zeroes.

7.3.2 <Path Latency> Parameter [RFC2210, RFC2215]



The Path Latency is a single 32-bit integer in network byte order. The composition rule for the <Path Latency> parameter is summation with a clamp of (2**32 - 1) on the maximum value. The latencies are average values reported in units of one microsecond. A system with resolution less than one microsecond MUST set unused digits to zero. An individual QNE can advertise a latency value between 1 and 2**28 (somewhat over two minutes) and the total latency added across all QNEs can range as high as (2**32)-2. If the sum of the different elements delays exceeds (2**32)-2, the end-to-end advertised delay SHOULD be reported as indeterminate. A QNE that cannot accurately predict the latency of packets it is processing MUST raise the not-supported flagand either leave the value of Path Latency as is, or add its best estimate of its lower bound. A raised not-supported flagflag indicates the value of Path Latency is a lower bound of the real Path Latency. The distinguished value (2**32)-1 is taken to mean indeterminate latency because the composition function limits the composed sum to this value, it indicates the range of the composition calculation was exceeded.

7.3.3 <Path Jitter> Parameter



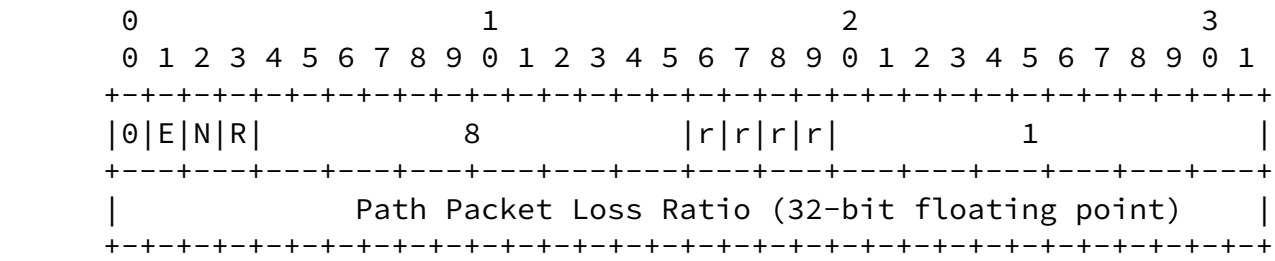
The Path Jitter is a set of four 32-bit integers in network byte order. The Path Jitter parameter is the combination of four

statistics describing the Jitter distribution with a clamp of (2**32 - 1) on the maximum of each value. The jitter STATs are reported in units of one microsecond. A system with resolution less than one microsecond MUST set unused digits to zero. An individual

QNE can advertise jitter values between 1 and 2**28 (somewhat over two minutes) and the total jitter computed across all QNEs can range as high as (2**32)-2. If the combination of the different element values exceeds (2**32)-2, the end-to-end advertised jitter SHOULD be reported as indeterminate. A QNE that cannot accurately predict the jitter of packets it is processing MUST raise the not-supported flag and either leave the value of Path Jitter as is, or add its best estimate of its STAT values. A raised not-supported flag indicates the value of Path Jitter is a lower bound of the real Path Jitter. The distinguished value (2**32)-1 is taken to mean indeterminate jitter. A QNE that cannot accurately predict the jitter of packets it is processing SHOULD set its local parameter to this value. Because the composition function limits the total to this value, receipt of this value at a network element or application indicates that the true path jitter is not known. This MAY happen because one or more network elements could not supply a value, or because the range of the composition calculation was exceeded.

NOTE: The Jitter composition function makes use of the <Path Latency> parameter. Composition functions for loss, latency and jitter may be found in [Y.1541].

7.3.4 <Path PLR> Parameter

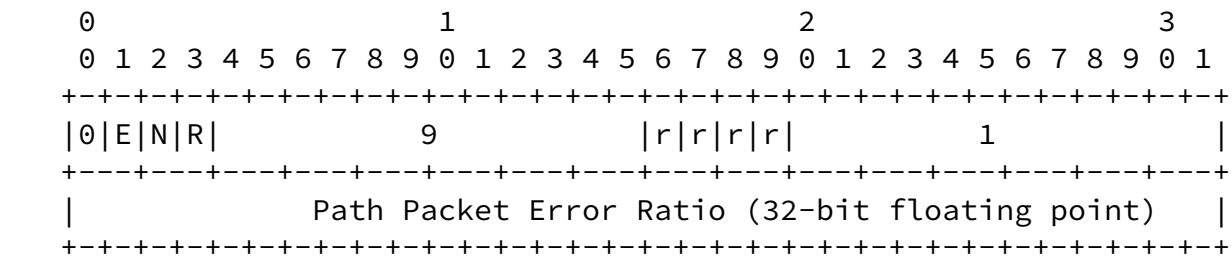


The Path PLR is a single 32-bit single precision IEEE floating point number in network byte order. The composition rule for the <Path PLR> parameter is summation with a clamp of 10^-1 on the maximum value. The PLRs are reported in units of 10^-11. A system with resolution less than one microsecond MUST set unused digits to zero. An individual QNE can advertise a PLR value between zero and 10^-2 and the total PLR added across all QNEs can range as high as 10^-1.

If the sum of the different elements values exceeds 10^-1, the end-to-end advertised PLR SHOULD be reported as indeterminate. A QNE that cannot accurately predict the PLR of packets it is processing MUST raise the not-supported flag and either leave the value of Path PLR as is, or add its best estimate of its lower bound. A raised not-supported flag indicates the value of Path PLR is a lower bound of the real Path PLR. The distinguished value 10^-1 is taken to mean indeterminate PLR. A QNE which cannot accurately predict the PLR of packets it is processing SHOULD set its local parameter to this value. Because the composition function limits the composed sum to this value, receipt of this value at a network element or application indicates that the true path PLR is not known. This MAY happen

because one or more network elements could not supply a value, or because the range of the composition calculation was exceeded.

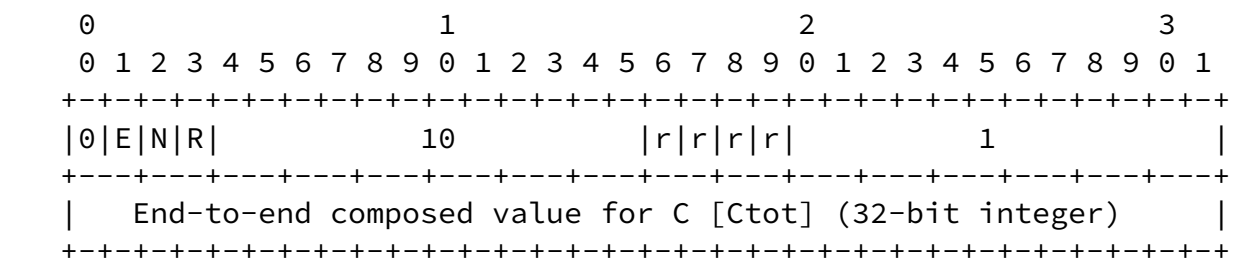
7.3.5 <Path PER> Parameter

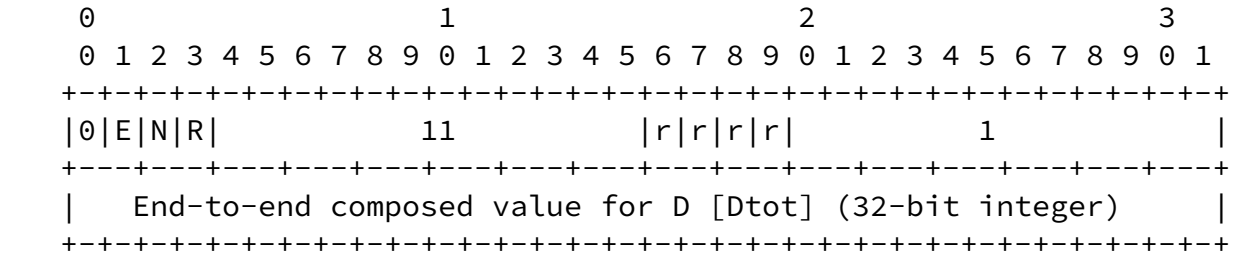


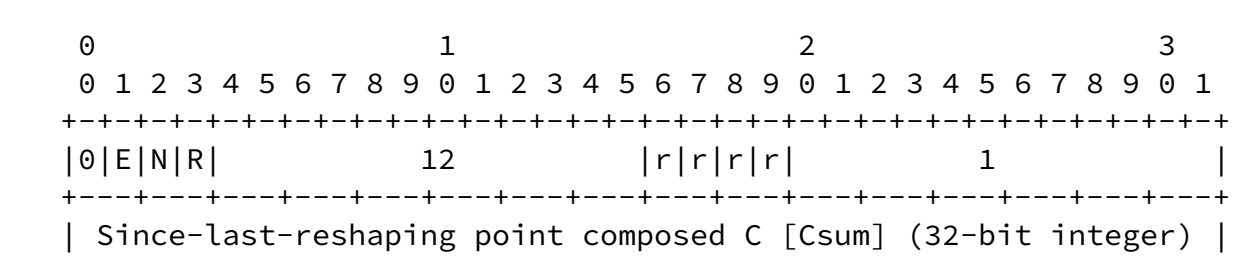
The Path PER is a single 32-bit single precision IEEE floating point number in network byte order. The composition rule for the <Path PER> parameter is summation with a clamp of 10^-1 on the maximum value. The PERs are reported in units of 10^-11. A system with resolution less than one microsecond MUST set unused digits to zero. An individual QNE can advertise a PER value between zero and 10^-2 and the total PER added across all QNEs can range as high as 10^-1. If the sum of the different elements values exceeds 10^-1, the end-to-end advertised PER SHOULD be reported as indeterminate. A QNE that cannot accurately predict the PER of packets it is processing MUST raise the not-supported flag and either leave the value of Path PER as is, or add its best estimate of its lower bound. A raised not-supported flag indicates the value of Path PER is a lower bound of the real Path PER. The distinguished value 10^-1 is taken to mean indeterminate PER. A QNE which cannot accurately predict the PER of packets it is processing SHOULD set its local parameter to this value. Because the composition function limits the composed sum to this value, receipt of this value at a network element or application

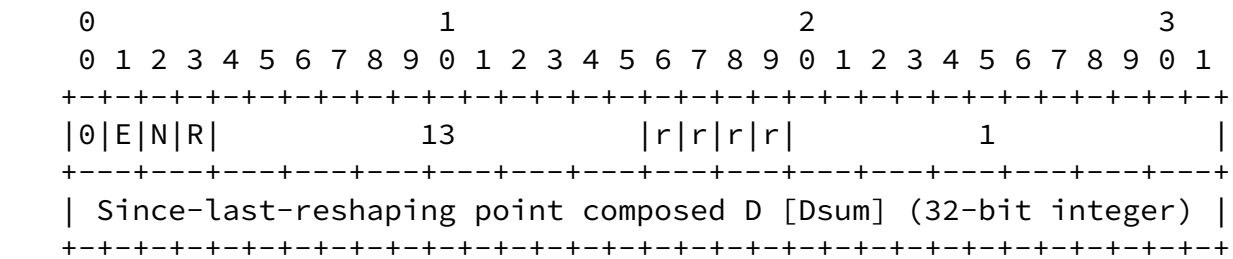
indicates that the true path PER is not known. This MAY happen because one or more network elements could not supply a value, or because the range of the composition calculation was exceeded.

7.3.6 <Ctot> <Dtot> <Csum> <Dsum> Parameters [RFC2210, RFC2212, RFC2215]





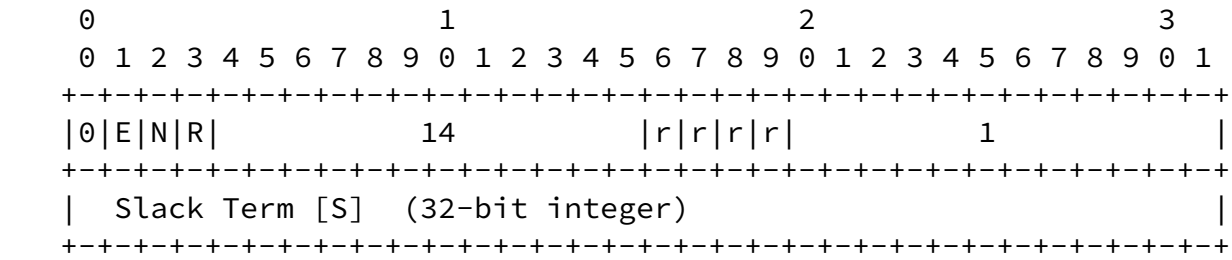




The error term C is measured in units of bytes. An individual QNE can advertise a C value between 1 and 2**28 (a little over 250 megabytes) and the total added over all QNEs can range as high as (2**32)-1. Should the sum of the different QNEs delay exceed

(2**32)-1, the end-to-end error term MUST be set to (2**32)-1. The error term D is measured in units of one microsecond. An individual QNE can advertise a delay value between 1 and 2**28 (somewhat over two minutes) and the total delay added over all QNEs can range as high as (2**32)-1. Should the sum of the different QNEs delay exceed (2**32)-1, the end-to-end delay MUST be set to (2**32)-1.

7.3.7 <Slack Term> Parameter [RFC2212, RFC2215]



Slack term S MUST be nonnegative and is measured in microseconds. The Slack term, S, is represented as a 32-bit integer. Its value can range from 0 to (2**32)-1 microseconds.

8. Security Considerations

The priority parameter raises possibilities for theft of service attacks because users could claim an emergency priority for their flows without real need, thereby effectively preventing serious emergency calls to get through. Several options exist for countering such attacks, for example

- only some user groups (e.g. the police) are authorized to set the emergency priority bit

- any user is authorized to employ the emergency priority bit for particular destination addresses (e.g. police)

9. IANA Considerations

This section defines the registries and initial codepoint assignments for the QSPEC template, in accordance with BCP 26 RFC 2434 [RFC2434]. It also defines the procedural requirements to be followed by IANA in allocating new codepoints.

This specification creates the following registries with the

structures as defined below:

Object Types (12 bits):
The following values are allocated by this specification:
0-4: assigned as specified in Section 7:
Object Type = 0: control information
= 1: QoS Desired
= 2: QoS Available
= 3: QoS Reserved
= 4: Minimum QoS
The allocation policies for further values are as follows:
5-63: Standards Action
64-127: Private/Experimental Use
128-4095: Reserved
(Note: 'Reserved' just means 'do not give these out'.)

QSPEC Version (4 bits):
The following value is allocated by this specification:
0: assigned to Version 0 QSPEC
The allocation policies for further values are as follows:
1-15: Standards Action
A specification is required to depreciate, delete, or modify QSPEC versions.

QOSM ID (12 bits):
The allocation policies are as follows:
0-63: Specification Required
64-127: Private/Experimental Use
128-4095: Reserved

A specification is required to depreciate, delete, or modify QOSM IDs.

QSPEC Procedure (8 bits):
Broken down into
Message Sequence (4 bits):
The following values are allocated by this specification:
0-2: assigned as specified in Section 7.1:
Message Sequence 0:
Semantic: QSPEC Procedure = Sender-Initiated Reservations
(see Section 6.4.1)
Message Sequence 1:
Semantic: QSPEC Procedure = Receiver-Initiated Reservations
(see Section 6.4.2)

Message Sequence 2:
Semantic: QSPEC Procedure = Resource Queries (see Section 6.4.3)
The allocation policies for further values are as follows:
3-15: Standards Action
Object Combination (4 bits):
The following values are allocated by this specification:
The Object Combination field can take the values between
1 and 3 indicated in the tables in Section 6:
Message Sequence: 0
Object Combination: 1, 2, 3
Semantic: see table in Section 6.4.1
Message Sequence: 1
Object Combination: 1, 2, 3
Semantic: see table in Section 6.4.2
Message Sequence: 2
Object Combination: 1, 2, 3
Semantic: see table in Section 6.4.3
The allocation policies for further values are as follows:
3-15: Standards Action
A specification is required to depreciate, delete, or modify QSPEC Procedures.

Error Code (16 bits)
The allocation policies are as follows:
0-127: Specification Required
128-255: Private/Experimental Use
255-65535: Reserved
A specification is required to depreciate, delete, or modify Error Codes.

Parameter ID (12 bits):
The following values are allocated by this specification:
1-14: assigned as specified in Sections 7.2 and 7.3:
Parameter ID 1: <Excess Treatment> Parameter
2: <Traffic> Parameter
3: <QoS Class> Parameter
4: <Priority> Parameter

5: <Token Bucket-2> Parameter
6: <Path Latency> Parameter
7: <Path Jitter> Parameter
8: <Path PLR> Parameter
9: <Path PER> Parameter
10: <Ctot> Parameter

```
                11: <Dtot> Parameter
                12: <Csum> Parameter
                13: <Dsum> Parameters
                14: <Slack Term> Parameter
   The allocation policies for further values are as follows:
   15-63: Standards Action (for QSPEC-1 parameters)
   64-127: Specification Required (for QSPEC-2 parameters)
   128-255: Private/Experimental Use
   255-4095: Reserved

   A specification is required to depreciate, delete, or modify
   Parameter IDs.  Note that if additional QSPEC-1 parameters are
   defined in the future, this requires a standards action equivalent to
   reissuing this document as a QSPEC-bis.

   Excess Treatment Parameter (8 bits):
   The following values are allocated by this specification:
   0-3: assigned as specified in Section 7.2.1:
   Excess Treatment Parameter 0: drop
                              1: shape
                              2: remark
                              3: no metering or policing is
                                 permitted
   The allocation policies for further values are as follows:
   4-63: Standards Action
   64-255: Reserved
   Remark Value (8 bits)
   The allocation policies are as follows:
   0-63: Specification Required
   64-127: Private/Experimental Use
   128-255: Reserved

   DSTE Class Type Sub-Parameter (8 bits):
   The following values are allocated by this specification:
   0-7: assigned as specified in Section 7.2.3:
   DSTE Class Type Sub-Parameter 0: DSTE Class Type 0
                                 1: DSTE Class Type 1
                                 2: DSTE Class Type 2
                                 3: DSTE Class Type 3
                                 4: DSTE Class Type 4
                                 5: DSTE Class Type 5
                                 6: DSTE Class Type 6
                                 7: DSTE Class Type 7
   The allocation policies for further values are as follows:
   8-63: Standards Action
```

64-255: Reserved

Y.1541 QoS Class Sub-Parameter (8 bits):
The following values are allocated by this specification:
0-7: assigned as specified in Section 7.2.3:
Y.1541 QoS Class Sub-Parameter 0: Y.1541 QoS Class 0
1: Y.1541 QoS Class 1
2: Y.1541 QoS Class 2
3: Y.1541 QoS Class 3
4: Y.1541 QoS Class 4
5: Y.1541 QoS Class 5
6: Y.1541 QoS Class 6
7: Y.1541 QoS Class 7
The allocation policies for further values are as follows:
8-63: Standards Action
64-255: Reserved

Admission Priority Parameter (8 bits):
The following values are allocated by this specification:
0-2: assigned as specified in Section 7.2.4:
Admission Priority 0: best-effort priority flow
1: normal priority flow
2: high priority flow
255: admission priority not used
The allocation policies for further values are as follows:
3-63: Standards Action
64-254: Reserved

RPH Namespace Parameter (16 bits):
Note that [RFC4412] creates a registry for RPH Namespace and Priority values already (see Section 12.6 of [RFC4412]). A QSPEC registry is also created because the assigned values are being mapped to QSPEC parameter values. The following values are allocated by this specification:
0-5: assigned as specified in Section 7.2.4:
The allocation policies for further values are as follows:
6-63: Standards Action
64-65535: Reserved

RPH Priority Parameter (8 bits):
dsn namespace:
The allocation policies are as follows:
0-63: Standards Action
64-255: Reserved
drsn namespace:
The allocation policies are as follows:
0-63: Standards Action
64-255: Reserved
Q735 namespace:

The following values are allocated by this specification:
0-4: assigned as specified in Section 7.2.4:

Q735 priority 4: q735.4
3: q735.3
2: q735.2
1: q735.1
0: q735.0
The allocation policies for further values are as follows:
5-63: Standards Action
64-255: Reserved
ets namespace:
The following values are allocated by this specification:
0-4: assigned as specified in Section 7.2.4:
ETS priority 4: ets.4
3: ets.3
2: ets.2
1: ets.1
0: ets.0
The allocation policies for further values are as follows:
5-63: Standards Action
64-255: Reserved
wts namespace:
The following values are allocated by this specification:
0-4: assigned as specified in Section 7.2.4:
WPS priority 4: wps.4
3: wps.3
2: wps.2
1: wps.1
0: wps.0
The allocation policies for further values are as follows:
5-63: Standards Action
64-255: Reserved

10. Acknowledgements

The authors would like to thank (in alphabetical order) David Black, Ken Carlberg, Anna Charny, Christian Dickman, Adrian Farrel, Ruediger Geib, Matthias Friedrich, Xiaoming Fu, Janet Gunn, Robert Hancock, Chris Lang, Jukka Manner, An Nguyen, Dave Oran, Tom Phelan, James Polk, Alexander Sayenko, John Rosenberg, Bernd Schloer, Hannes Tschofenig, and Sven van den Bosch for their very helpful suggestions.

11. Normative References

[3GPP-1] 3GPP TS 22.067 V7.0.0 (2006-03) Technical Specification, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; enhanced Multi Level Precedence and Preemption service (eMLPP) - Stage 1 (Release 7).
[3GPP-2] 3GPP TS 23.067 V7.1.0 (2006-03) Technical Specification, 3rd Generation Partnership Project; Technical Specification Group Core Network; enhanced Multi-Level Precedence and Preemption service (eMLPP) - Stage 2 (Release 7).

[3GPP-3] 3GPP TS 24.067 V6.0.0 (2004-12) Technical Specification, 3rd Generation Partnership Project; Technical Specification Group Core Network; enhanced Multi-Level Precedence and Preemption service (eMLPP) - Stage 3 (Release 6).
[DSCP-REGISTRY] http://www.iana.org/assignments/dscp-registry
[PHBID-CODES-REGISTRY] http://www.iana.org/assignments/phbid-codes
[GIST] Schulzrinne, H., Hancock, R., "GIST: General Internet Signaling Transport," work in progress.
[QoS-SIG] Manner, J., et. al., "NSLP for Quality-of-Service Signaling," work in progress.
[RFC1832] Srinivasan, R., "XDR: External Data Representation Standard," RFC 1832, August 1995.
[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC2210] Wroclawski, J., "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997.
[RFC2212] Shenker, S., et. al., "Specification of Guaranteed Quality of Service," September 1997.
[RFC2215] Shenker, S., Wroclawski, J., "General Characterization Parameters for Integrated Service Network Elements", RFC 2215, Sept. 1997.
[RFC2475] Blake, S., et. al., "An Architecture for Differentiated Services", RFC 2475, December 1998.
[RFC3140] Black, D., et. al., "Per Hop Behavior Identification Codes," June 2001.
[RFC3181] Herzog, S., "Signaled Preemption Priority Policy Element," RFC 3181, October 2001.
[RFC3290] Bernet, Y., et. al., "An Informal Management Model for Diffserv Routers," RFC 3290, May 2002.
[RFC4124] Le Faucheur, F., et. al., "Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering," RFC 4124, June 2005.
[RFC4412] Schulzrinne, H., Polk, J., "Communications Resource Priority for the Session Initiation Protocol(SIP)," RFC 4412,

February 2006.
[Y.1541] ITU-T Recommendation Y.1541, "Network Performance Objectives for IP-Based Services," May 2002.
[Y.1571] ITU-T Recommendation Y.1571, "Admission Control Priority Levels in Next Generation Networks," July 2006.

12. Informative References

[DQOS] Cablelabs, "PacketCable Dynamic Quality of Service Specification," CableLabs Specification PKT-SP-DQOS-I12-050812, August 2005.
[IEEE754] Institute of Electrical and Electronics Engineers, "IEEE Standard for Binary Floating-Point Arithmetic," ANSI/IEEE Standard 754-1985, August 1985.
[CL-QOSM] Kappler, C., "A QoS Model for Signaling IntServ Controlled-Load Service with NSIS," work in progress.
[NETWORK-BYTE-ORDER] Wikipedia, "Endianness," http://en.wikipedia.org/wiki/Endianness.

[NSIS-EXTENSIBILITY] Loughney, J., "NSIS Extensibility Model", work in progress.
[Q.2630] ITU-T Recommendation Q.2630.3: "AAL Type 2 Signaling Protocol - Capability Set 3" Sep. 2003
[RFC2205] Braden, B., et. al., "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification," RFC 2205, September 1997.
[RFC2434] Narten, T., Alvestrand, H., "Guidelines for Writing an IANA Considerations Section in RFCs," RFC 3181, October 1998.
[RFC2474] Nichols, K., et. al., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474, December 1998.
[RFC2597] Heinanen, J., et. al., "Assured Forwarding PHB Group," RFC 2597, June 1999.
[RFC2997] Bernet, Y., et. al., "Specification of the Null Service Type," RFC 2997, November 2000.
[RFC3140] Black, D., et. al., "Per Hop Behavior Identification Codes," RFC 3140, June 2001.
[RFC3393] Demichelis, C., Chimento, P., "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM), RFC 3393, November 2002.
[RFC3564] Le Faucheur, F., et. al., Requirements for Support of Differentiated Services-aware MPLS Traffic Engineering, RFC 3564, July 2003
[RFC3726] Brunner, M., et. al., "Requirements for Signaling Protocols", RFC 3726, April 2004.
[RMD-QOSM] Bader, A., et. al., "RMD-QOSM - The Resource Management

in Diffserv QOS Model," work in progress.
[VERTICAL-INTERFACE] Dolly, M., Tarapore, P., Sayers, S., "Discussion on Associating of Control Signaling Messages with Media Priority Levels," T1S1.7 & PRQC, October 2004.
[Y.1540] ITU-T Recommendation Y.1540, "Internet Protocol Data Communication Service - IP Packet Transfer and Availability Performance Parameters," December 2002.
[Y.1541-QOSM] Ash, J., et. al., "Y.1541-QOSM -- Y.1541 QoS Model for Networks Using Y.1541 QoS Classes," work in progress.

13. Authors' Addresses

Jerry Ash (Editor)
AT&T
Room MT D5-2A01
200 Laurel Avenue
Middletown, NJ 07748, USA
Phone: +1-(732)-420-4578
Fax: +1-(732)-368-8659
Email: gash@att.com

Attila Bader (Editor)
Traffic Lab
Ericsson Research
Ericsson Hungary Ltd.
Laborc u. 1 H-1037

Budapest Hungary
Email: Attila.Bader@ericsson.com

Cornelia Kappler (Editor)
Siemens AG
Siemensdamm 62
Berlin 13627
Germany
Email: cornelia.kappler@siemens.com

Appendix A. Example of QSPEC Processing

This appendix illustrates the operation and use of the QSPEC within the NSLP. The example configuration in shown in Figure 3.

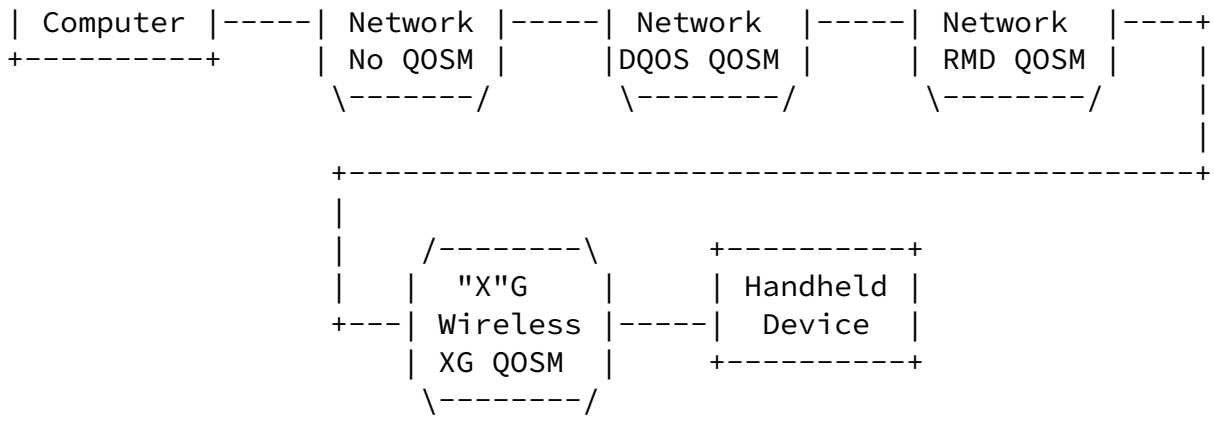


Figure 3: Example Configuration to Illustrate QoS-NSLP/QSPEC Operation

In this configuration, a laptop computer and a handheld wireless device are the endpoints for some application that has QoS requirements. Assume initially that the two endpoints are stationary during the application session, later we consider mobile endpoints. For this session, the laptop computer is connected to a home network that has no QoS support. The home network is connected to a CableLabs-type cable access network with dynamic QoS (DQOS) support, such as specified in the 'CMS to CMS Signaling Specification' [DQOS] for cable access networks. That network is connected to a DiffServ core network that uses the RMD QOSM [RMD-QOSM]. On the other side of the DiffServ core is a wireless access network built on generation "X" technology with QoS support as defined by generation "X". And finally the handheld endpoint is connected to the wireless access network.

We assume that the Laptop is the QNI and handheld device is the QNR.

The QNI will signal an initiator QSPEC object to achieve the QoS desired on the path. As stated in Section 4, the QNI MUST support

at least one QOSM, but it may not know the QOSM supported by the network. In any case, if the QNI supports only one QOSM, it would normally signal a reservation according to the requirements of that QOSM. Furthermore, the QNI would most likely support the QOSM that matches its functionality. For example, the default QOSM for mobile phones might be the XG-QOSM, while the CL-QOSM might be the default for workstations.

Referring to Figure 3, the laptop computer may choose the

CL-QOSM because it is connected to a wired network. If the handheld device acts as the QNI, it may choose the XG-QOSM because it is connected to the XG wireless network. On the other hand, a particular QOSM could be configured if a user/administrator knows that some particular QOSM is used. For example, if the laptop computer is connected to the XG network via the XG phone, which acts as a modem, then it reasonable to specify the XG-QOSM since resources are accessed through the XG network,

In this example we consider two different ways to perform sender-initiated signaling for QoS:

Case 1) The QNI sets <QoS Desired>, <QoS Available> and possibly <Minimum QoS> QSPEC objects in the initiator QSPEC, and initializes <QoS Available> to <QoS Desired>. Since this is a reservation in a heterogenic network with different QOSMs supported in different domains, each QNE on the path reads and interprets those parameters in the initiator QSPEC that it needs to implement the QOSM within its domain (as described below). Each QNE along the path checks to see if <QoS Available> resources can be reserved, and if not, the QNE reduces the respective parameter values in <QoS Available> and reserves these values. The minimum parameter values are given in <Minimum QoS>, if populated, otherwise zero if <Minimum QoS> is not included. If one or more parameters in <Available QoS> fails to satisfy the corresponding minimum values in Minimum QoS, the QNE generates a RESPONSE message to the QNI and the reservation is aborted. Otherwise, the QNR generates a RESPONSE to the QNI with the <QoS Available> for the reservation.

Case 2) The QNI signals the initiator QSPEC with <QoS Desired>. Since this is a reservation in a heterogenic network with different QOSMs supported in different domains, each QNE on the path reads and interprets those parameters in the initiator QSPEC that it needs to implement the QOSM within its domain (as described below). If a QNE cannot reserve <QoS Desired> resources, the reservation fails.

In both cases, the QNI populates QSPEC-1 and QSPEC-2 to ensure correct treatment of its traffic in domains down the path. Since the QNI does not know the QOSM used in downstream domains, it includes values for those QSPEC-1 and QSPEC-2 parameters consistent with the QOSM it is signaling and any additional parameters it cares about. Let us assume the QNI wants to achieve

IntServ-like QoS guarantees, and also is interested in what path

latency it can achieve. The QNI therefore includes in the QSPEC the QOSM ID for IntServ Controlled Load Service. The QSPEC objects are signaled with all parameters necessary for IntServ Controlled Load and additionally the parameter to measure path latency, as follows:

<QoS Desired> = <Token Bucket>
<QoS Available> = <Token Bucket> <Path Latency>

Additionally, to ensure correct treatment further down the path, the QNI may include <QoS Class> in <QoS Desired>.

In both cases, each QNE on the path reads and interprets those parameters in the initiator QSPEC that it needs to implement the QOSM within its domain. It may need additional parameters for its QOSM, which are not specified in the initiator QSPEC. If possible, these parameters must be inferred from those that are present, according to rules defined in the QOSM implemented by this QNE.

There are three possibilities when a RESERVE message is received at a QNE at a domain border (we illustrate these possibilities in the example):

- the QNE just leaves the QSPEC as-is.

- the QNE can stack a local QSPEC on top of the initiator QSPEC (this is new in QoS NSLP, RSVP does not do this).

- the QNE can tunnel the initiator RESERVE message through its domain and issue its own local RESERVE message. For this new local RESERVE message, the QNE acts as the QNI, and the QSPEC in the domain is an initiator QSPEC. This procedure is also used by RSVP in making aggregate reservations, in which case there is not a new intra-domain (aggregate) RESERVE for each newly arriving interdomain (per-flow) RESERVE, but the aggregate reservation is updated by the border QNE (QNI) as need be. This is also how RMD works [RMD-QOSM].

For example, at the RMD domain, a local RESERVE with its own RMD initiator QSPEC corresponding to the RMD-QOSM is generated based on the original initiator QSPEC according to the procedures described in Section 4.5 of [QoS-SIG] and in [RMD-QOSM]. That is, the ingress QNE to the RMD domain must map the QSPEC parameters contained in the original initiator QSPEC into the RMD QSPEC. The RMD QSPEC for example needs <Bandwidth> and <QoS Class>. <Bandwidth> is generated from the <Token Bucket> parameter. Information on <QoS Class>, however, is not provided. According to the rules laid out in the RMD QOSM, the ingress QNE infers from the fact that an IntServ Controlled Load QOSM was signaled that the EF PHB is appropriate to set the <PHB Class> parameter. These RMD QSPEC parameters are populated in the RMD initiator QSPEC generated within the RMD domain.

Furthermore, the node at the egress to the RMD domain updates <QoS Available> on behalf of the entire RMD domain if it can. If it cannot, it raises the parameter-specific, 'not-supported' flag, warning the QNR that the final value of these parameters in QoS Available is imprecise.

In the XG domain, the initiator QSPEC is translated into a local QSPEC using a similar procedure as described above. The local QSPEC becomes the current QSPEC used within the XG domain, that is, the it becomes the first QSPEC on the stack, and the initiator QSPEC is second. This saves the QNEs within the XG domain the trouble of re-translating the initiator QSPEC. At the egress edge of the XG domain, the translated local QSPEC is popped, and the initiator QSPEC returns to the number one position.

If the reservation was successful, eventually the RESERVE request arrives at the QNR (otherwise the QNE at which the reservation failed would have aborted the RESERVE and sent an error RESPONSE back to the QNI). If the RII was included in the QoS NSLP message, the QNR generates a positive RESPONSE with QSPEC objects <QoS Reserved> - and for case 1 - additionally <QoS Available>. The parameters appearing in <QoS Reserved> are the same as in <QoS Desired>, with values copied from <QoS Available> in case 1, and with the original values from <QoS Desired> in case 2. That is, it is not necessary to transport the <QoS Desired> object back to the QNI since the QNI knows what it signaled originally, and the information is not useful for QNEs in the reverse direction. The <QoS Reserved> object should transport all necessary information, although the <QoS Available> and <QoS Reserved> objects may end up transporting some of the same information.

Hence, the QNR includes the following QSPEC objects in the RESPONSE:

<QoS Reserved> = <Token Bucket>
<QoS Available> = <Token Bucket> <Path Latency>

If the handheld device on the right of Figure 3 is mobile, and moves through different "XG" wireless networks, then the QoS might change on the path since different XG wireless networks might support different QOSMs. As a result, QoS NSLP/QSPEC processing will have to renegotiate the <QoS Available> on the path. From a QSPEC perspective, this is like a new reservation on the new section of the

path and is basically the same as any other rerouting event - to the QNEs on the new path it looks like a new reservation. That is, in this mobile scenario, the new segment may support a different QOSM than the old segment, and the QNI would now signal a new reservation (explicitly, or implicitly with the next refreshing RESERVE message) to account for the different QOSM in the XG wireless domain. Further details on rerouting are specified in [QoS-SIG].

For bit-level examples of QSPECs see the documents specifying QOSMs [CL-QOSM, Y.1541-QOSM, RMD-QOSM].

Appendix B. Mapping of QoS Desired, QoS Available and QoS Reserved of NSIS onto AdSpec, TSpec and RSpec of RSVP IntServ

The union of QoS Desired, QoS Available and QoS Reserved can provide all functionality of the objects specified in RSVP IntServ, however it is difficult to provide an exact mapping.

In RSVP, the Sender TSpec specifies the traffic an application is going to send (e.g. token bucket). The AdSpec can collect path characteristics (e.g. delay). Both are issued by the sender. The receiver sends the FlowSpec which includes a Receiver TSpec describing the resources reserved using the same parameters as the Sender TSpec, as well as a RSpec which provides additional IntServ QoS Model specific parameters, e.g. Rate and Slack.

The RSVP TSpec/AdSpec/RSpec seem quite tailored to receiver-initiated signaling employed by RSVP, and the IntServ QoS Model. E.g. to the knowledge of the authors it is not possible for the sender to specify a desired maximum delay except implicitly and mutably by seeding the AdSpec accordingly. Likewise, the RSpec is only meaningfully sent in the receiver-issued RSVP RESERVE message. For this reason our discussion at this point leads us to a slightly different mapping of necessary functionality to objects, which should result in more flexible signaling models.

Appendix C. Change History & Open Issues

C.1 Change History (since Version -04)

Version -05:

- fixed <QOSM hops> in Sec. 5 and 6.2 as discussed at Interim Meeting

- discarded QSPEC parameter <M> (Maximum packet size) since MTU discovery is expected to be handled by procedure currently defined by PMTUD WG
- added "container QSPEC parameter" in Sec. 6.1 to augment encoding efficiency
- added the 'tunneled QSPEC parameter flag' to Sections 5 and 6
- revised Section 6.2.2 on SIP priorities
- added QSPEC procedures for "RSVP-style reservation", resource queries and bidirectional reservations in Sec. 7.1
- reworked Section 7.2

Version -06:

- defined "not-supported flag" and "tunneled parameter flag" (subsumes "QSPEC-2 parameter flag")
- defined "error flag" for error handling

- updated bit error rate (BER) parameter to packet loss ratio (PLR) parameter
- added packet error ratio (PER) parameter
- coding checked by independent expert
- coding updated to include RE flags in QSPEC objects and MENT flags in QSPEC parameters

Version -07:

- added text (from David Black) on DiffServ QSPEC example in Section 6
- re-numbered QSPEC parameter IDs to start with 0 (Section 7)
- expanded IANA Considerations Section 9

Version -08:

- update to 'RSVP-style' reservation in Section 6.1.2 to mirror what is done in RSVP
- modified text (from David Black) on DiffServ QSPEC example in Section 6.2
- update to general QSPEC parameter formats in Section 7.1 (length restrictions, etc.)
- re-numbered QSPEC parameter IDs in Section 7.2
- modified <Excess Treatment> parameter values in Section 7.2.2
- update to reservation priority Section 7.2.7
- specify the 3 "STATS" in the <Path Jitter> parameter, Section 7.2.9.4

- minor updates to IANA Considerations Section 9

Version -09:

- remove the DiffServ example in Section 6.2 (intent is use text as a basis for a separate DIFFSERV-QOSM I-D)
- update wording in example in Section 4.3, to reflect use of default QOSM and QOSM selection by QNI
- make minor changes to Section 7.2.7.2, per the exchange on the list
- add comment on error codes, after the first paragraph in Section 4.5.1

Version -10:

- rewrote Section 2.0 for clarity
- added clarifications on QSPEC-1 parameters in Section 4.2; added discussion of forwarding options when a domain supports a different QOSM than the QNI
- expanded Section 4.5 on error code handling, including redefined E-Flag and editorial changes to the N-Flag and T-Flag discussions
- made some editorial clarifications in Section 4.6 on defining new mandatory (QSPEC-1) parameters, and also reference the [NSIS-EXTENSIBILITY] document
- Section 4.7 added to identify what a QOSM specification document

---

must include
- clarified the requirements in Section 5.0 for defining a new QSPEC Version
- made editorial changes to Section 6, and added procedures for handling preemption
- removed QOSM ID assignments in Section 9.0; clarified procedures for defining new QSPEC-1 parameters; added registry of QOSM error codes

Version -11:

- 'QSPEC-1 parameter' replaces 'mandatory QSPEC parameter'
- 'QSPEC-2 parameter' replaces 'optional QSPEC parameter'
- R-flag ('remapped parameter flag') introduced to denote remapping, approximating, or otherwise not strictly interpreting a QSPEC parameter
- T-flag ('tunneled parameter flag') eliminated and incorporated within the R-flag
- Section 4 rewritten on QOSM concept, QSPEC processing, etc. to

provide a more logical flow of information
- read-write/read-only flag associated with QSPEC objects eliminated and object itself defined as rw or ro without a flag
- <Non QOSM Hop> parameter redefined as non-QOSM-hop Q-flag
- Section 7 on QSPEC parameter definitions revised to clearly separate QSPEC-1 parameter coding from QSPEC-2 parameter coding
- <Traffic>, <QoS Class>, and <Priority> QSPEC-1 parameters mapped to container parameters
- references updated to include normative references defining procedures to 'strictly interpret' each QSPEC parameter
- Section 7.2.1 on PHB class updated to specify additional RFC 3140 cases
- Section 7.2.1 on excess treatment updated to specify excess treatment behaviors

Version -12:

- Sections 4, 5, 6: Many editorial changes and rearrangements
- Moved example of QSPEC processing to Appendix A
- Section 7.2.2: Changed <Traffic Parameter> to fixed length Parameter
- 3 open issues in version -11 resolved without change

C.2 Open Issues

None.

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in

this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at

http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement