IETF Internet Draft NSIS Working Group
Internet Draft
Intended status: Informational
<draft-ietf-nsis-qspec-19.txt>
Expiration Date: August 2008

G. Ash
AT&T
A. Bader
Ericsson
C. Kappler
deZem GmbH
D. Oran
Cisco Systems, Inc.
February 25, 2008

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 including a number of QSPEC parameters. The QSPEC parameters provide a common language to be re-used in several QOSMs and thereby aim to ensure the extensibility and interoperability of QoS NSLP. The node initiating the NSIS signaling adds an initiator

QSPEC, which indicates the QSPEC parameters that must be interpreted by the downstream nodes less the reservation fails, 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. Introduction

The QoS NSIS signaling layer protocol (NSLP) [QoS-SIG] is used to signal QoS reservations for a data flow, provide forwarding resources (QoS) for that flow, and establish and maintain state at nodes along the path of the flow. The design of QoS NSLP is conceptually similar to the decoupling between RSVP [RFC2205] and the IntServ architecture [RFC2210], where a distinction is made between the operation of the signaling protocol and the information required for the operation of the Resource Management Function (RMF). [QoS-SIG] describes the signaling protocol, while this document describes the RMF-related information carried in the QSPEC (QoS Specification) object carried in QoS NSLP messages.

[QoS-SIG] defines four QoS NSLP messages - RESERVE, QUERY, RESPONSE, and NOTIFY - each of which may carry the QSPEC object, while this document describes a template for the QSPEC object. The QSPEC object carries information on traffic descriptions, resources required, resources available, and other information required by the RMF. Therefore the QSPEC template described in this document is closely tied to QoS NSLP and the reader should to be familiar with [QoS-SIG] to fully understand this document.

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, and defines the behavior of the RMF that reserves resources for each flow, including inputs and outputs. The QoS NSLP protocol is able to signal QoS

reservations for different QOSMs, wherein all information specific to a QOSM is encapsulated in the QSPEC object and only the RMF specific to a given QOSM will need to interpret the QSPEC. Examples of QOSMs are IntServ, DiffServ admission control, and those specified in

[Y.1541-QOSM, CL-QOSM, RMD-QOSM].

QSPEC parameters include, for example, a mandatory traffic model (TMOD) parameter, constraints parameters, such as path latency and path jitter, traffic handling directives, such as excess treatment, and traffic classifiers such as PHB class.

QSPEC objects loosely correspond to the TSpec, RSpec and AdSpec objects specified in RSVP and may contain, respectively, a description of QoS desired, QoS reserved, and QoS available. Going beyond RSVP functionality, the QSPEC 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, as defined in [QoS-SIG]. 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 is enhanced by the definition of a common set of QSPEC parameters. A QoS NSIS initiator (QNI) initiating the QoS NSLP signaling adds an initiator QSPEC object containing parameters describing the desired QoS, normally based on the QOSM it supports. QSPEC parameters flagged by the QNI must be interpreted by all QNEs in the path, else the reservation fails. In contrast, QSPEC parameters not flagged by the QNI may be skipped if not understood. Additional QSPEC parameters can be defined by informational specification documents, and thereby ensure the extensibility and flexibility of QoS NSLP.

A local QSPEC can be defined in a local domain with the initiator QSPEC encapsulated, where the local QSPEC must be functionally consistent with the initiator QSPEC in terms of defined source traffic and other constraints. That is, a domain specific local QSPEC can be defined and processed in a local domain, which could, for example, can enable simpler processing by QNEs within the local domain.

In Section 3.4 a worked example of QSPEC processing is provided.

2. Terminology

Initiator QSPEC: The initiator QSPEC is included into a QoS NSLP message by the QNI/QNR. It travels end-to-end to the QNR/QNI and is never removed.

Local QSPEC: A local QSPEC is used in a local domain and is domain specific. It encapsulates the initiator QSPEC and is

removed at the egress of the local domain.
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 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; specifies what sub-set of QSPEC QoS constraints & traffic handling directives a QNE implementing that QOSM is capable of supporting & how resources will be managed by the RMF.

QoS Reserved: QSPEC object containing parameters describing the reserved resources and related QoS parameters.

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

QSPEC parameter: Any parameter appearing in a QSPEC; for example, traffic model (TMOD), path latency, and excess treatment parameters.

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

QSPEC Type: Identifies a particular QOSM used in the QSPEC

Resource Management Function (RMF): Functions that are related to resource management and processing of QSPEC parameters.

3. QSPEC 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 informational 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.

3.1 QoS Models

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. The RMF implements functions that are related to resource management and processes the QSPEC parameters.

QOSMs affect the operation of the RMF in NSIS-capable nodes, the information carried in QSPEC objects, and may under some circumstances (e.g. aggregation) cause a separate NSLP session to be instantiated by having the RMF as a QNI. QOSM specifications may define RMF triggers that cause the QoS NSLP to run semantics within the underlying QoS NSLP signaling state and messaging processing rules, as defined in Section 5.2 of [QoS-SIG]. New QoS NSLP message processing rules can only be defined in Standards Track extensions to QoS NSLP. If a QOSM specification defines triggers that deviate from existing standard QoS NSLP processing rules (must be standards track in that case), the fallback for QNEs not supporting that QOSM are the standard QoS NSLP state transition/message processing rules.

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 parameters it uses to describe the desired QoS and resource control in the RMF, and it may define additional QSPEC parameters.

When a QoS NSLP message travels through different domains, it may encounter different QOSMs. Since QOSM use different QSPEC parameters for describing resources, the QSPEC parameters included by the QNI may not be understood in other domains. The QNI therefore can flag those QSPEC parameters it considers vital with the M flag. QSPEC parameters with the M flag set must be interpreted by the downstream QNEs, or the reservation fails. QSPEC parameters without the M flag set should be interpreted by the downstream QNEs, but may be ignored if not understood.

A QOSM specification MUST include the following:
- role of QNEs, e.g., location, frequency, statefulness, etc.
- QSPEC definition including QSPEC parameters
- QSPEC procedures applicable to this QOSM
- QNE processing rules describing how QSPEC information is treated and interpreted in the RMF, 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 parameters the QOSM specification defines
- define what happens in case of preemption if the default QNI behavior (tear down preempted reservation) is not followed (see

Section 4.3.5)

A QOSM specification MAY include the following:

- define additional QOSM-specific error codes, as discussed in Section 4.2.3
- can state which QoS-NSLP options a QOSM wants to use, when several options are available for a QOSM (e.g., local QSPEC to either a) hide initiator QSPEC within a local domain message, or b) encapsulate initiator QSPEC).

QOSMs are free, subject to IANA registration and review rules, to extend QSPECs by adding parameters of any of the kinds supported by the standard QSPEC. This includes traffic description parameters, constraint parameters and traffic handling directives. QOSMs are not permitted, however, to reinterpret or redefine the standard QSPEC parameters specified in this document. Note that signaling functionality is only defined by the QoS NSLP document [QoS-SIG] and not by this document or by QOSM specification documents.

3.2 QSPEC Objects

The QSPEC is the object of QoS NSLP containing QSPEC objects and parameters. QSPEC objects are the main building blocks of the QSPEC parameter set that is input or output of an RMF operation. QSPEC parameters are the parameters appearing in a QSPEC, which must include traffic (TMOD), and may optionally include constraints (e.g., path latency), traffic handling directives (e.g., excess treatment), and traffic classifiers (e.g., PHB class). The RMF implements functions that are related to resource management and processes the QSPEC parameters.

The QSPEC consists of a QSPEC version number and QSPEC objects. IANA assigns a new QSPEC version number when changes that are not backwards compatible are made to the QSPEC and this document is reissued. Note that a new QSPEC version number is not needed when new QSPEC parameters are specified. Later QSPEC versions MUST be backward compatible with earlier QSPEC versions. That is, a version n+1 device must support QSPEC version n (or earlier). On the other hand, if a QSPEC version n (or earlier) device receives an NSLP message specifying QSPEC version n+1, then the version n device responds with an 'Incompatible QSPEC' error code (0x0f) response, as discussed in Section 4.2.3, allowing the QNE that sent the NSLP message to retry with a lower QSPEC version.

This document provides a template for the QSPEC in order to promote interoperability between QOSMs. Figure 1 illustrates how the QSPEC is composed of up to four QSPEC objects, namely QoS Desired, QoS

Available, QoS Reserved and Minimum QoS. Each of these QSPEC objects consists of a number of QSPEC parameters. A given QSPEC may contain only a subset of the QSPEC objects, e.g. QoS Desired. The QSPEC objects QoS Desired, QoS Available and QoS Reserved MUST be supported

by QNEs. Minimum QoS MAY be supported.

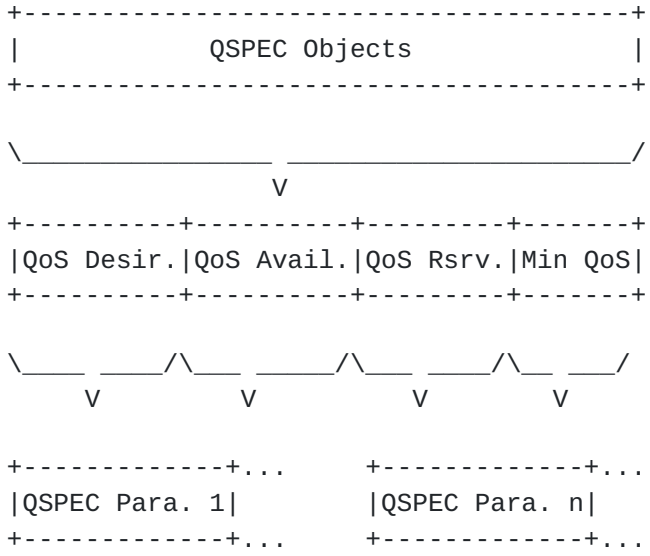


Figure 1: Structure of the QSPEC

The QoS Desired Object describe the resources the QNI desires to reserve and hence this is a read-only QSPEC object in that the QSPEC parameters carried in the object may not be overwritten. QoS Desired is always included in a RESERVE message (see [QoS-SIG] for descriptions of the QoS NSLP RESERVE, QUERY, RESPONSE, and NOTIFY messages).

The QoS Available Object travels in a RESERVE or QUERY message and collects information on the resources currently available on the path. Hence QoS Available in this case is a read-write object, which means the QSPEC parameters contained in QoS Available may be updated, but they cannot be deleted). 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.

QoS Reserved reflects the resources that were reserved. It is a read-only 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.

Parameters cannot be overwritten in this 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.

Note that the relationship of QSPEC objects to RSVP objects is covered in Appendix A.

3.3 QSPEC Parameters

QSPEC parameters provide a common language for building QSPEC objects. This document defines a number of QSPEC parameters, additional parameters may be defined in separate QOSM specification documents. For example, QSPEC parameters are defined in [RMD-QOSM] and [Y.1541-QOSM].

One QSPEC parameter, <TMOD>, is special. It provides a description of the traffic for which resources are reserved. This parameter must be included by the QNI and it must be interpreted by all QNEs. All other QSPEC parameters are populated by a QNI if they are applicable to the underlying QoS desired. For these QSPEC parameters, the QNI sets the M flag if they must be interpreted by downstream QNEs. If QNEs cannot interpret the parameter the reservation fails. QSPEC parameters populated by a QNI without the M flag set should be interpreted by downstream QNEs, but may be ignored if not understood.

In this document the term 'interpret' means, in relation to RMF processing of QSPEC parameters, that the RMF processes the QSPEC parameter according to the commonly accepted normative procedures specified by references given for each QSPEC parameter. Note that a QNE need only interpret a QSPEC parameter if it is populated in the QSPEC object by the QNI; if not populated in the QSPEC, the QNE does not interpret it of course.

Note that when an ingress QNE in a local domain defines a local QSPEC and encapsulates the initiator QSPEC, the QNEs in the interior local domain need only process the local QSPEC and can ignore the initiator (encapsulated) QSPEC. However, edge QNEs in the local domain indeed must interpret the QSPEC parameters populated in the initiator QSPEC with the M flag set and should interpret QSPEC parameters populated in the initiator QSPEC without the M flag set

As described in the previous section, QoS parameters may be overwritten depending on which QSPEC object, and which message, they appear in.

3.3.1 Traffic Model Parameter

The <Traffic Model> (TMOD) parameter is mandatory for the QNI to include in the initiator QSPEC and mandatory for downstream QNEs to interpret The traffic description specified by the TMOD parameter is a container consisting of 4 sub-parameters:

- o rate (r)
- o bucket size (b)
- o peak rate (p)
- o minimum policed unit (m)

All 4 of the sub-parameters MUST be included in the TMOD parameter. The TMOD parameter can be set to describe the traffic source. If, for example, TMOD is set to specify bandwidth only, then set r = peak rate = p, b = large, m = large. As another example if TMOD is set for TCP traffic, then set r = average rate, b = large, p = large.

When the 4 TMOD sub-parameters are included in QoS Available, they provide information, for example, about the TMOD resources available along the path followed by a data flow. The value of TMOD at a QNE is an estimate of the TMOD resources the QNE has available for packets following the path up to the next QNE, including its outgoing link, if this link exists. Furthermore, the QNI MUST account for the resources of the ingress link, if this link exists. 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, as well as physical resources.

The output composed value is the minimum of the QNE's value and the input composed value for r, b, and p, and the maximum of the QNE's value and the input composed value for m. This quantity, when composed end-to-end, informs the QNR (or QNI in a RESPONSE message) of the minimal TMOD resources along the path from QNI to QNR.

Two TMOD parameters are defined in Section 5, <TMOD-1> and <TMOD-2>, where the second (<TMOD-2>) parameter is specified as could be needed to support some DiffServ applications. For example, it is typically assumed that DiffServ EF traffic is shaped at the ingress by a single rate token bucket. Therefore, a single TMOD parameter is sufficient to signal DiffServ EF traffic. However, for DiffServ AF traffic two sets of token bucket parameters are needed, one token bucket for the average traffic and one token bucket for the burst traffic. [RFC2697] defines a Single Rate Three Color Marker (srTCM), which meters a traffic stream and marks its packets according to three traffic parameters, Committed Information Rate (CIR), Committed Burst Size (CBS), and Excess Burst Size (EBS), to be either green, yellow,

or red. A packet is marked green if it does not exceed the CBS, yellow if it does exceed the CBS, but not the EBS, and red otherwise. [RFC2697] defines specific procedures using two token buckets that run at the same rate. Therefore 2 TMOD parameters are sufficient to

distinguish among 3 levels of drop precedence. An example is also described in the Appendix to [RFC2597].

3.3.2 Constraints Parameters

<Path Latency>, <Path Jitter>, <Path PLR>, and <Path PER> are QSPEC 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.

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 the combination of speed-of-light propagation delay, packet processing, and queueing. 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 ratio (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 ratio (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.

The slack term parameter 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].

3.3.3 Traffic Handling Directives

An application MAY like to reserve resources for packets and also specify a specific traffic handling behavior, such as <Excess Treatment>. In addition, as discussed in Section 3.1, an application MAY like to define RMF triggers that cause the QoS NSLP to run semantics within the underlying QoS NSLP signaling state / messaging processing rules, as defined in Section 5.2 of [QoS-SIG]. Note, however, that new QoS NSLP message processing rules can only be defined in Standards Track extensions to the QoS NSLP.
As with constraints parameters and other QSPEC parameters, traffic-handling-directives parameters may be defined in QOSM specifications in order to provide support for QOSM-specific resource management functions. Such QOSM-specific parameters are already defined, for example, in [RMD-QOSM], [CL-QOSM] and [Y.1541-QOSM].

Generally, a traffic-handling-directives parameters is expected to be set by the QNI in <QoS Desired>, and to not be included in

<QoS Available>. If such a parameter is included in <QoS Available>, QNEs may change their value.

The <Preemption Priority> parameter 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. The <Defending Priority> parameter 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.

The <Excess Treatment> parameter describes how the QNE will process out-of-profile traffic. Excess traffic MAY be dropped, shaped and/or remarked.

3.3.4 Traffic Classifiers

An application MAY like to reserve resources for packets with a particular DiffServ per-hop behavior (PHB) [RFC2475]. Note that PHB class is normally set by a downstream QNE to tell the QNI how to mark traffic to ensure treatment committed by admission control, however, setting of the parameter by the QNI is not precluded. An application MAY like to reserve resources for packets with a particular QoS class, e.g. Y.1541 QoS class [Y.1541] or DiffServ-aware MPLS traffic engineering (DSTE) class type [RFC3564, RFC4124]. These parameters are useful in various QOSMs, e.g., [RMD-QOSM], [Y.1541-QOSM], and other QOSMs yet to be defined (e.g., DSTE-QOSM). This is intended to provide guidelines to QOSMs on how to encode these parameters; use of the PHB class parameter is illustrated in the example in the following section.

3.4 Example of QSPEC Processing

This section illustrates the operation and use of the QSPEC within the NSLP. The example configuration in shown in Figure 2.

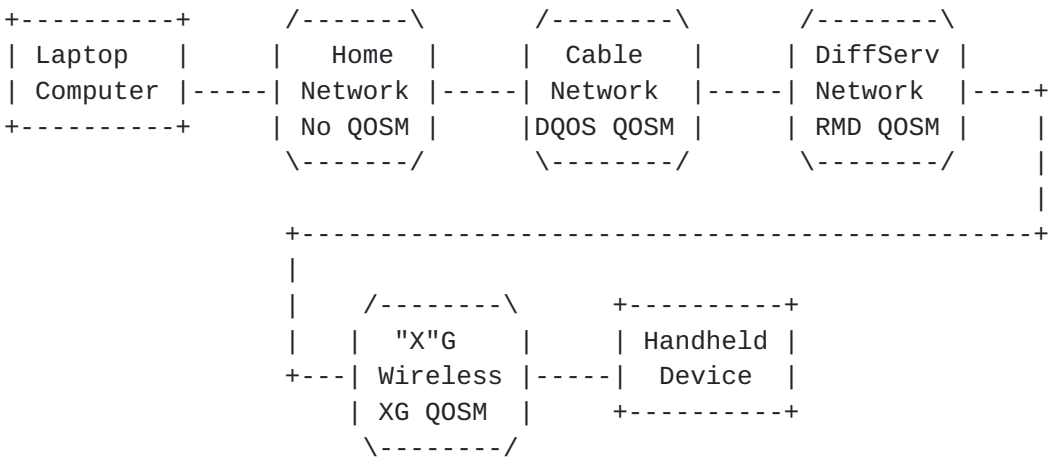


Figure 2: Example Configuration to Illustrate QoS-NSLP/QSPEC Operation

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 [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.

The QNI sets QoS Desired, QoS Available and possibly Minimum QoS QSPEC objects in the initiator QSPEC, and initializes QoS Available to QoS Desired. Each QNE on the path reads and interprets those parameters in the initiator QSPEC and checks to see if QoS Available resources can be reserved. 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 QoS Available 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. If a QNE cannot reserve QoS Desired resources, the reservation fails.

The QNI populates QSPEC parameters to ensure correct treatment of its traffic in domains down the path. Let us assume the QNI wants to achieve IntServ-Controlled Load-like QoS guarantees, and also is

interested in what path latency it can achieve. Additionally, to ensure correct treatment further down the path, the QNI includes <PHB Class> in <QoS Desired>. The QNI therefore includes in the QSPEC

QoS Desired = <TMOD> <PHB Class>
QoS Available = <TMOD> <Path Latency>

Since <Path Latency> and <QoS Class> are not vital parameters from the QNI's perspective, it does not raise their M flags.

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 add a local QSPEC and encapsulate the initiator QSPEC (see discussion in Section 4.1; this is new in QoS NSLP, RSVP does not do this).

- the QNE can 'hide' the initiator RESERVE message so that only the edge QNE processes the initiator RESERVE message, which then bypasses intermediate nodes between the edges of the domain, and issues its own local RESERVE message (see Section 3.3.1 of [QoS-SIG]). For this new local RESERVE message, the QNE acts as the QNI, and the QSPEC in the domain is an initiator QSPEC. A similar 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]. The ingress QNE to the RMD domain maps the TMOD parameters contained in the original initiator QSPEC into the equivalent TMOD parameter representing only the peak bandwidth in the local QSPEC. The local RMD QSPEC for example also needs <PHB Class>, which in this case was provided by the QNI.

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 (since the M flag is not set for <Path Latency>) it raises the parameter-specific, 'not-supported' flag, warning the QNR that the final latency value 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, and the initiator QSPEC is encapsulated. This saves the QNEs within the XG

domain the trouble of re-translating the initiator QSPEC, and simplifies processing in the local domain. At the egress edge of the XG domain, the translated local QSPEC is removed, 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 QoS Available. The parameters appearing in QoS Reserved are the same as in QoS Desired, with values copied from QoS Available. Hence, the QNR includes the following QSPEC objects in the RESPONSE:

QoS Reserved = <TMOD> <PHB Class>
QoS Available = <TMOD> <Path Latency>

If the handheld device on the right of Figure 2 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].

4. QSPEC Processing & Procedures

Three flags are used in QSPEC processing, the M flag, E flag, and N flag, which are explained in this section. The QNI sets the M flag for each QSPEC parameter it populates that must be interpreted by downstream QNEs. If a QNE does not support parameter it sets the N flag and fails the reservation. If the QNE supports the parameter but cannot meet the resources requested by the parameter, it sets the E flag and fails the reservation.

If the M flag is not set, the downstream QNE SHOULD interpret the parameter. If the QNE does not support the parameter it sets the N flag and forwards the reservation. If the QNE supports the parameter but cannot meet the resources requested by the parameter,

it sets the E flag and fails the reservation.

4.1 Local QSPEC Definition & Processing

A QNE at the edge of a local domain may either a) translate the initiator QSPEC into a local QSPEC and encapsulate the initiator QSPEC in the RESERVE message, or b) 'hiding' 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.

A local QSPEC may allow a simpler control plane in a local domain. The edge nodes in the local domain must interpret the initiator QSPEC parameters. They can either initiate a parallel session with local QSPEC or define a local QSPEC and encapsulate the initiator QSPEC, as illustrated in Figure 3. The Initiator/Local QSPEC bit identifies whether the QSPEC is an initiator QSPEC or a local QSPEC. The QSPEC Type indicates, for example, that the initiator of local QSPEC uses to a certain QOSM, e.g., CL-QSPEC Type. It may be useful for the QNI to signal a QSPEC Type based on some QOSM (which will necessarily entail populating certain QOSM-related parameters) so that a downstream QNE can chose amongst various QOSM-related processes it might have. That is, the QNI populates the QSPEC Type, e.g., CL-QSPEC Type and sets the Initiator/Local QSPEC bit to 'Initiator'. A local QNE can decide, for whatever reasons, to Insert a local QSPEC Type, e.g., RMD-QSPEC Type, and set the local QSPEC Type = RMD-QSPEC and set the Initiator/Local QSPEC bit to 'Local' (and encapsulate the Initiator QSPEC in the RESERVE or whatever NSLP message).

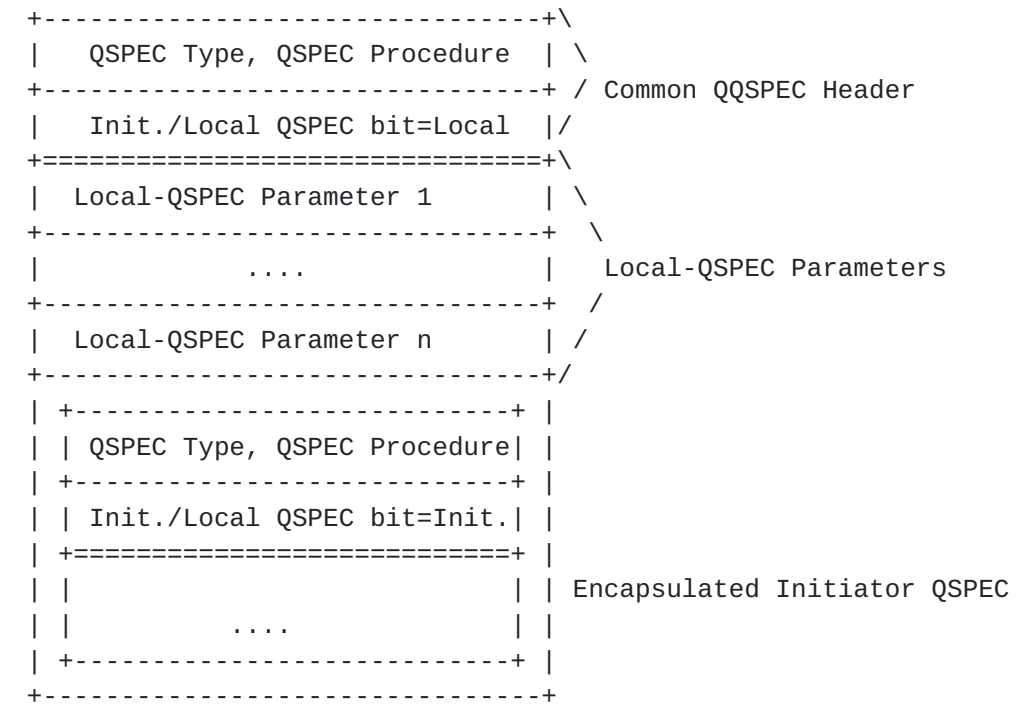


Figure 3. Defining a Local QSPEC

Here the QoS-NSLP only sees and passes one QSPEC up to the RMF. The type of the QSPEC thus may change within a local domain. Hence

- o the QNI signals its QoS requirements with the initiator QSPEC,
- o the ingress edge QNE in the local domain translates the initiator QSPEC parameters to equivalent parameters in the local QSPEC,
- o the QNEs in the local domain only interpret the local QSPEC parameters
- o the egress QNE in the local domain processes the local QSPEC and also interprets the QSPEC parameters in the initiator QSPEC.

The local QSPEC MUST be consistent with the initiator QSPEC. That is, it MUST NOT specify a lower level of resources than specified by the initiator QSPEC. For example, in RMD the TMOD parameters contained in the original initiator QSPEC are mapped into the equivalent TMOD parameter representing only the peak bandwidth in the local QSPEC.

Note that it is possible to use both a) hiding a QSPEC through a local domain by initiating a new RESERVE at the domain edge, and b) defining a local QSPEC and encapsulating the initiator QSPEC, as defined above. However, it is not expected that both the hiding and encapsulating functions would be use at the same time for the same flow.

The support of local QSPECs is a new and quite powerful capability, which is illustrated in Figure 4 for a single flow to show where the initiator and local QSPECs are used. The QNI initiates an end-to-end, inter-domain QoS NSLP RESERVE message containing the Initiator QSPEC for the Y.1541 QOSM. As illustrated in Figure 4, the RESERVE message crosses multiple domains supporting different QOSMs. In this illustration, the initiator QSPEC arrives in an QoS NSLP RESERVE message at the ingress node of the local-QOSM domain. At the ingress edge node of the local-QOSM domain, the end-to-end, inter-domain QoS-NSLP message triggers the generation of a local QSPEC, and the initiator QSPEC is encapsulated within the messages signaled through the local domain. The local QSPEC is used for QoS processing in the local-QOSM domain, and the initiator QSPEC is used for QoS processing outside the local domain.

In this example the QNI sets <Desired QoS>, <Minimum QoS>, <Available QoS> objects to include objectives for the <Path Latency>, <Path Jitter>, and <Path BER> parameters. The QNE/local domain sets the cumulative parameters, e.g., <Path Latency>, that can be achieved in the <QoS Available> object (but not less than specified in <Minimum QoS>). If the <Available QoS> fails to satisfy one or more of the <Minimum QoS> objectives, the QNE/local domain notifies the QNI and the reservation is aborted. If any QNE cannot meet the requirements designated by the initiator QSPEC to support a QSPEC parameter with the M bit set to zero, the QNE sets the N flag for

that parameter to one. Otherwise, the QNR notifies the QNI of the <QoS Available> for the reservation.

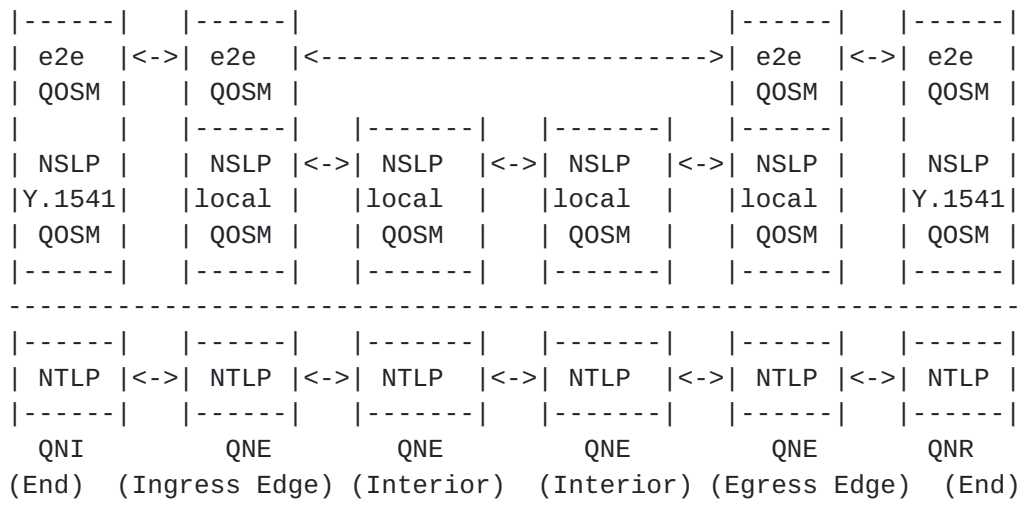


Figure 4. Example of Initiator & Local Domain QOSM Operation

4.2 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 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.

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 5.
- 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.

4.2.1 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 the TMOD parameter and all QSPEC parameters with the M flag set MUST be examined by the RMF, and all QSPEC parameters with the M flag not set SHOULD be examined by the RMF, and appropriately flagged. Additionally, the E flag in the corresponding QSPEC object MUST be raised when a resource cannot be satisfied for this parameter. 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.

4.2.2 QSPEC Parameter Not Supported N Flag

Each QSPEC 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 parameter. A QNE MUST set the not supported N flag if it cannot interpret the QSPEC parameter. If the M flag for the parameter is not set, 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 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 indicates that the cumulative value only provides a bound, and the QNI/QNR decides whether or not to accept the reservation with the N flag set.

4.2.3 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 4.3 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. In this case no E Flag is set, however one or more N flags may be set

- INFO_SPEC error class 0x04 (Transient Failure) / 0x08 (Reservation Failure):
  This code is set when at least one QSPEC parameter could not be satisfied, or when a QSPEC parameter with M flag set could not be interpreted. E flags are set for the parameters that could not be satisfied up to the QNE issuing the RESPONSE message. The N flag is set for those parameters that could not be interpreted by at least one QNE. 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 QNE where the error was found rejects the reservation.

- INFO_SPEC error class 0x03 (Protocol Error) / 0x0f (Incompatible QSPEC):
  A higher version QSPEC is signaled and not support by the QNE.

- INFO_SPEC error class 0x06 (QoS Model Error):
  QOSM error codes can be defined by QOSM specification documents. A registry is defined in Section 7 IANA Considerations.

4.2.4 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 4.3.

- 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.

4.2.5 Special Case of Local QSPEC

When an unsuccessful reservation problem occurs inside a local domain where a local QSPEC is used, only the topmost (local) QSPEC is affected (e.g. E flags are raised, etc.). The encapsulated initiator QSPEC is untouched. When the message (RESPONSE in case of stateful QNEs, RESERVE in case of stateless QNEs) however reaches the edge of the local domain, the local QSPEC is removed. The edge QNE must update the initiator QSPEC on behalf of the entire domain,

reflecting the information received in the local QSPEC. This update concerns both parameter values and flags. Note that some intelligence is needed in mapping the E flags, etc. from the local

QSPEC to the initiator QSPEC. For example, there may be no direct match between the parameters in the local and initiator QSPECs, but that does not mean that no E flags should be raised in the latter.

4.3 QSPEC Procedures

While the QSPEC template aims to put minimal restrictions on usage of QSPEC objects, 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. A positive RESPONSE message will only be generated if the QNE includes an RII (Request Identification Information) in the RESERVE message, and a negative RESPONSE message is always generated in case of an error or failure. 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.

4.3.1 Two-Way Transactions

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.

Table 1. Three Cases for Two-Way Transactions

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 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 from the QoS Available object (in RESERVE) into the QoS Reserved object (in RESPONSE). 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 can be overwritten in the RESERVE message, whereas they cannot be overwritten 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 <Traffic Handling Directives>, 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 <Traffic Handling Directives> parameters are read-only. Note that a QOSM specification may define its own <Traffic Handling Directives> parameters and processing rules.

4.3.2 Three-Way Transactions

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.

Table 2. Three Cases for Three-Way Transactions

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 3:

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 can be overwritten in the QUERY message, whereas they are cannot be overwritten 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.

The sender includes all QSPEC parameters it cares about in the QUERY message. Parameters that can be overwritten are updated by QNEs as the QUERY message travels towards the receiver. The receiver includes all QSPEC parameters arriving in the QUERY message also in the RESERVE message, with the value they had when arriving at the receiver. Again, QOSM-specific QSPEC 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.

Regarding <Traffic Handling Directives>, 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 <Traffic Handling Directives> parameters are read-only. Note that a QOSM specification may define its own <Traffic Handling Directives> parameters and processing rules.

4.3.3 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

Table 3. One Case for Two-Way Transactions

Note that the QoS Available object when traveling in the QUERY message can be overwritten, whereas in the RESPONSE message cannot be overwritten.

Regarding <Traffic Handling Directives>, 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 <Traffic Handling Directives> parameters are read-only. Note that a QOSM specification may define its own <Traffic Handling Directives> parameters and processing rules.

4.3.4 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.

4.3.5 Preemption

A flow can be preempted by a QNE based on QNE policy, where a decision to preempt a flow may account for various factors such as, for example, the values of the QSPEC preemption priority and defending priority parameters as described in Section 5.2.8. 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]. The NOTIFY message carries an INFO_SPEC with the error code as described in [QoS-SIG]. A QOSM specification document may specify whether a NOTIFY message also carries a QSPEC object. 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.

4.4 QSPEC Extensibility

Additional QSPEC parameters MAY need to be defined in the future and are defined in separate informational documents. For example, QSPEC 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 7 (IANA Considerations).

5. QSPEC Functional Specification

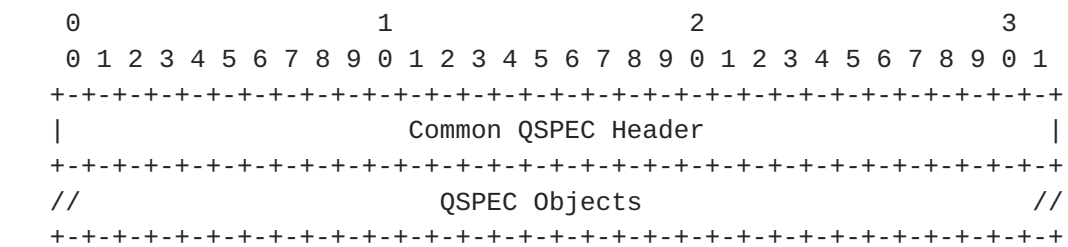
This section defines the encodings of the QSPEC parameters. We first give the general QSPEC formats and then the formats of the QSPEC objects and parameters.

Network byte order ('big-endian') for all 16- and 32-bit integers, as well as 32-bit floating point numbers, are as specified in [RFC4506, IEEE754, NETWORK-BYTE-ORDER].

5.1 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:

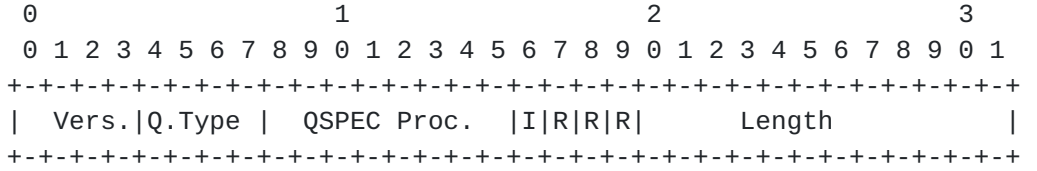
- The overall format is Type-Length-Value (in that order).
- Some parts of the type field are set aside for control flags.
- Length has the units of 32-bit words, and measures the length of Value. If there is no Value, Length=0. The Object length excludes the header.
- Value is a whole number of 32-bit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



5.1.1 Common Header Format

The Common QSPEC Header is a fixed 4-byte long object containing the QSPEC Version, QSPEC Type, an identifier for the QSPEC Procedure (see

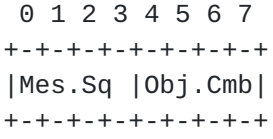
Section 4.3), and an Initiator/Local QSPEC bit:



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

QSPEC Type: Identifies the particular type of QSPEC, e.g., a QSPEC Type corresponding to a particular QOSM.

QSPEC Proc.: Identifies the QSPEC procedure and is composed of two times 4 bits. The first field identifies the Message Sequence, the second field identifies the QSPEC Object Combination used for this particular message sequence:



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 4.3:
Message Sequence: 0
Object Combination: 1, 2, 3
Semantic: see Table 1 in Section 4.3.1
Message Sequence: 1
Object Combination: 1, 2, 3
Semantic: see Table 2 in Section 4.3.2
Message Sequence: 2
Object Combination: 1
Semantic: see Table 3 in Section 4.3.3

I: Initiator/Local QSPEC bit identifies whether the QSPEC is an initiator QSPEC or a local QSPEC, and is set to the following values:

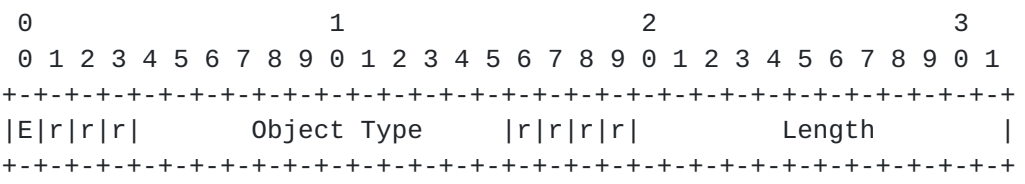
0: Initiator QSPEC
1: Local QSPEC

Length: The total length of the QSPEC excluding the common header

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.

5.1.2 QSPEC Object Header Format

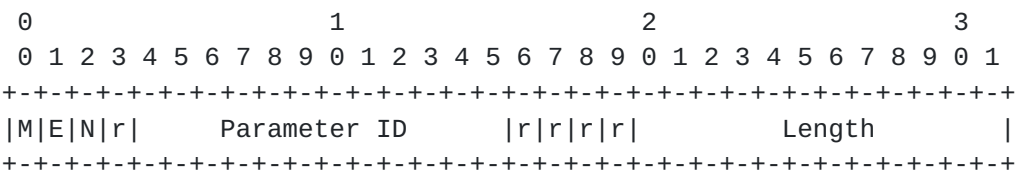
QSPEC objects share a common header format:



E Flag: Set if an error occurs on object level
Object Type = 0: QoS Desired (parameters cannot be overwritten)
= 1: QoS Available (parameters may be overwritten; see Section 3.3)
= 2: QoS Reserved (parameters cannot be overwritten)
= 3: Minimum QoS (parameters cannot be overwritten)

The r bits are reserved.

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



M Flag: When set indicates the subsequent parameter MUST be interpreted. Otherwise the parameter 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 4.2.1).
N Flag: Not-supported QSPEC parameter flag (see Section 4.2.2).
Parameter ID: Assigned to each parameter (see below)

Parameters are usually coded individually, for example, the <Excess Treatment> parameter (Section 5.2.11). 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 TMOD, 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, and N flags refer to the container. Examples of container parameters are <TMOD> (specified

below) and the PHR container parameter specified in [RMD-QOSM].

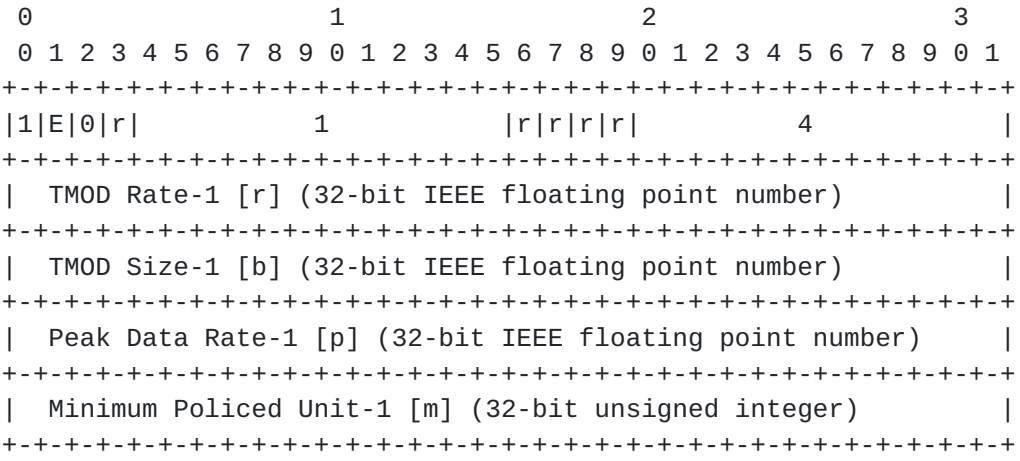
5.2 QSPEC Parameter Coding

The references in the following sections point to the normative procedures for processing the QSPEC parameters and sub-parameters.

5.2.1 <TMOD-1> Parameter [RFC2210, RFC2215]

The <TMOD-1> parameter consists of the <r>, <b>, <p>, and <m> sub-parameters, which all must be populated in the <TMOD-1> parameter. Note that a second TMOD QSPEC parameter <TMOD-2> is specified below in Section 5.2.2.

The coding for the <TMOD-1> parameter is as follows:

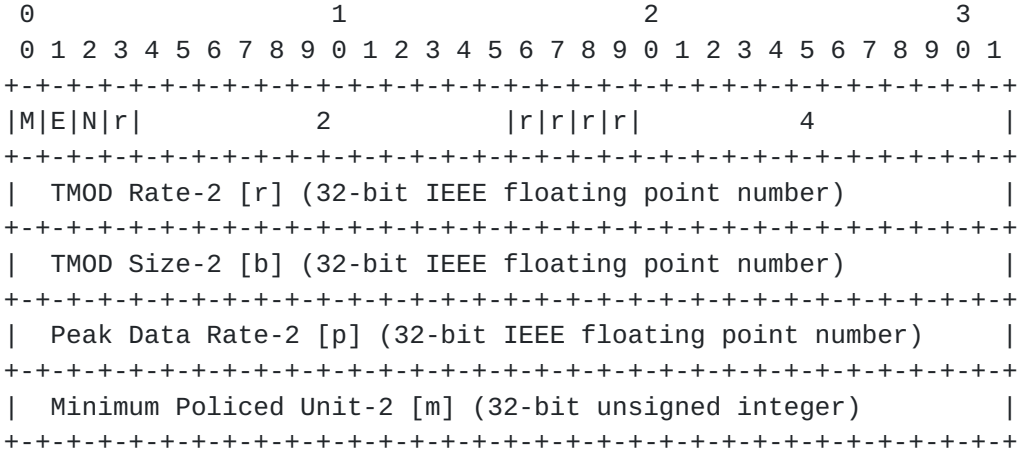


The <TMOD> parameters are represented by three floating point numbers in single-precision IEEE floating point format followed by one 32-bit integer in network byte order. The first floating point value is the rate (r), the second floating point value is the bucket size (b), the third floating point is the peak rate (p), and the first unsigned integer is the minimum policed unit (m). The values of r and p are measured in bytes/second, b and m are measured in bytes. 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.

5.2.2 <TMOD-2> Parameter [RFC2215]

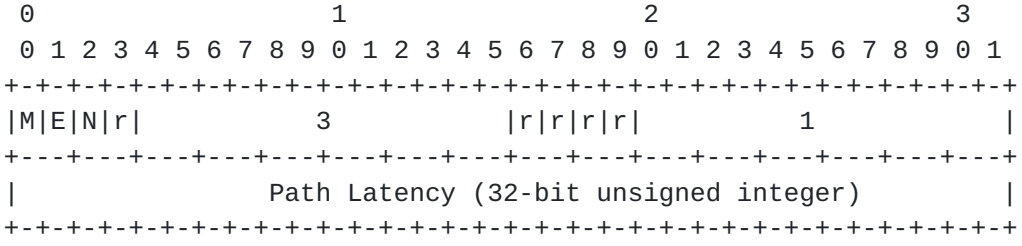
A second QSPEC <TMOD-2> parameter is specified as could be needed for example to support some DiffServ applications.

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.

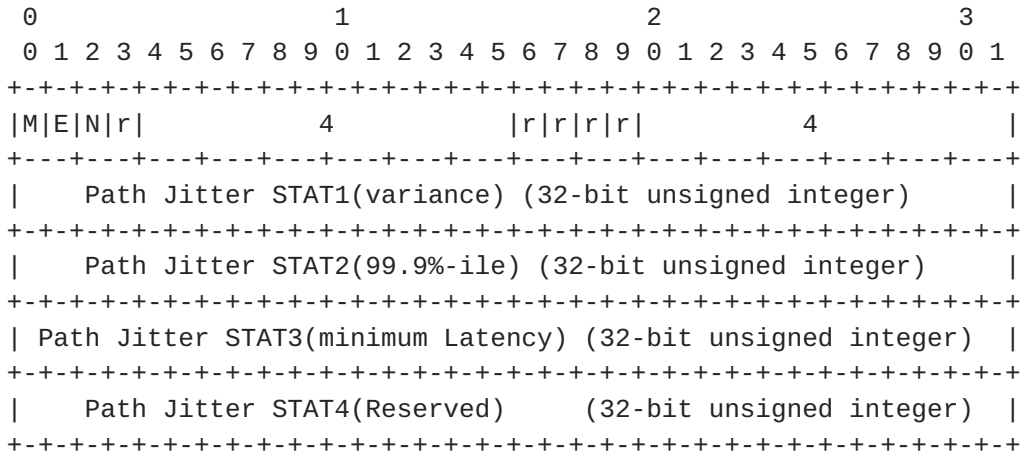
5.2.3 <Path Latency> Parameter [RFC2210, RFC2215]



The Path Latency is a single 32-bit unsigned 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 = (2**32)-1. A QNE that cannot accurately predict the latency of packets it is processing MUST raise the not-supported flag and either leave the value of Path Latency as is, or add its best estimate of its lower bound. A raised not-supported flag 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.

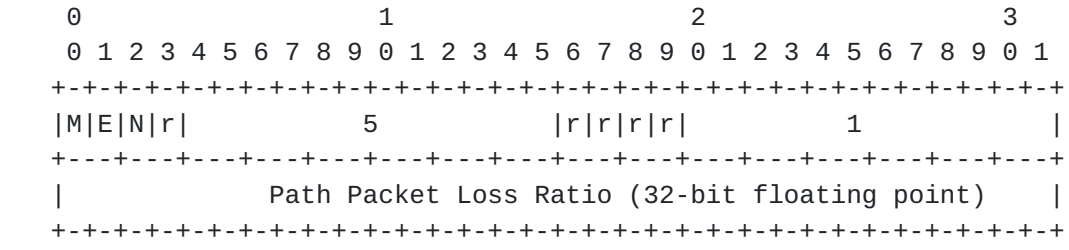
5.2.4 <Path Jitter> Parameter



The Path Jitter is a set of four 32-bit unsigned 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 path jitter 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.

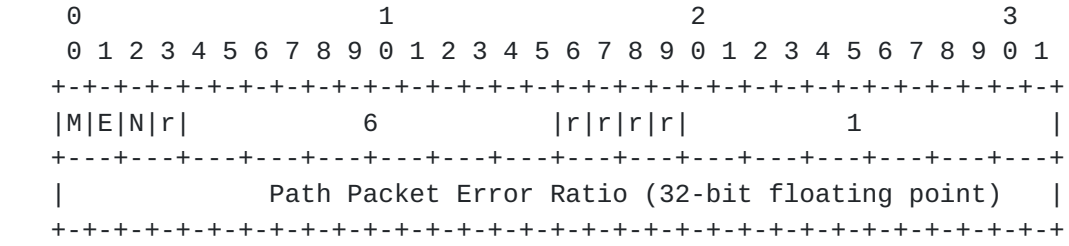
NOTE: The Jitter composition function makes use of the <Path Latency> parameter. Composition functions for loss, latency and jitter may be found in [Y.1541].

5.2.5 <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 10^-11 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 path PLR 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.

5.2.6 <Path PER> Parameter

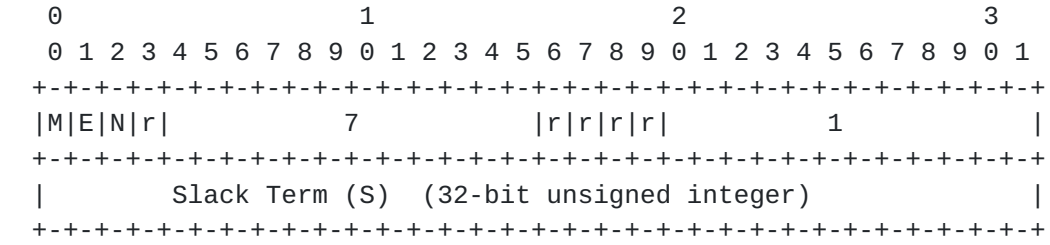


The Path PER is a single 32-bit single precision IEEE floating point number in network byte order. The composition rule for the <Path PER> parameter is summation with a clamp of 10^-1 on the maximum value. The PERs are reported in units of 10^-11. A system with resolution less than 10^-11 MUST set unused 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 path PER 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.

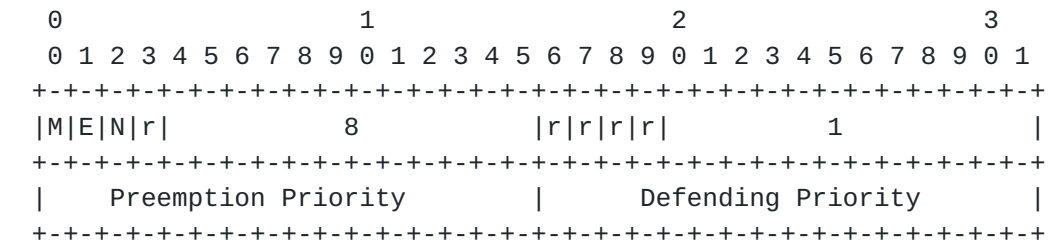
5.2.7 <Slack Term> Parameter [RFC2212, RFC2215]



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

5.2.8 <Preemption Priority> & <Defending Priority> Parameters [RFC3181]

The coding for the <Preemption Priority> & <Defending Priority> sub-parameters is as follows:



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

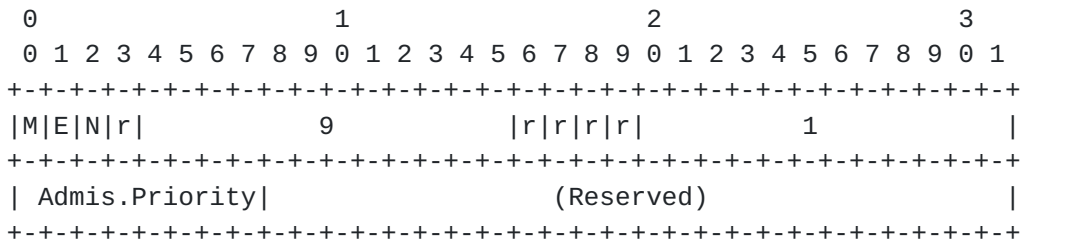
Defending Priority: Once a flow is admitted, the preemption priority becomes irrelevant. Instead, its defending priority is used to compare with the preemption priority of new flows.

As specified in [RFC3181], <Preemption Priority> and <Defending

Priority> are 16-bit integer values and both MUST be populated if the parameter is used.

5.2.9 <Admission Priority> Parameter [Y.2171]

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



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.2171], as follows:

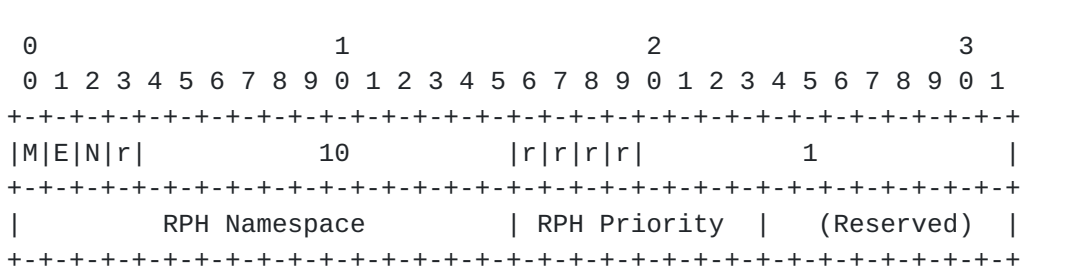
Admission Priority:

0 - best-effort priority flow
1 - normal priority flow
2 - high priority flow

A reservation without an <Admission Priority> parameter MUST be treated as a reservation with an <Admission Priority> = 1.

5.2.10 <RPH Priority> Parameter [RFC4412]

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



[RFC4412] defines a resource priority header (RPH) with parameters "RPH Namespace" and "RPH Priority", and if populated is applicable only to flows with high admission priority. A registry is created in [RFC4412] and extended in [EMERGENCY-RSVP] for IANA to assign the RPH priority parameter. In the extended registry, "Namespace Numerical Values" are assigned by IANA to RPH Namespaces and "Priority Numerical Values" are assigned to the RPH Priority.

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 than 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].

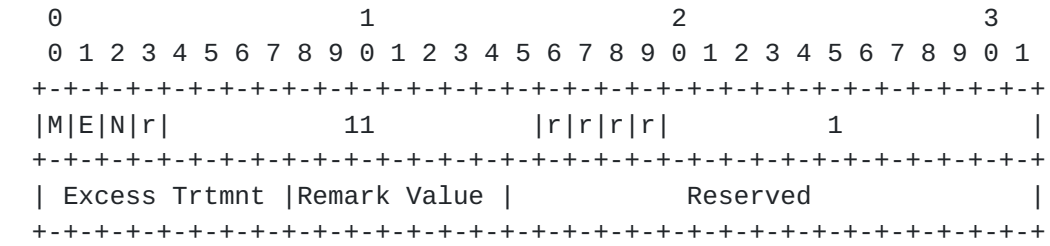
For the 4 priority parameters, the following cases are permissible (procedures specified in references):

1 parameter: <Admission Priority> [Y.2171]
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 parameters: <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.

5.2.11 <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. 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 TMOD parameter. Excess traffic is to be shaped to this TMOD. When the shaping causes unbounded queue growth at the shaper traffic can be dropped. If excess treatment is set to 'shape' and no TMOD parameter is given, the E flag is set for the parameter and the reservation fails.

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 or remarked to pertain to a particular QoS class (DSCP value). In the latter case, remarking relates to a DiffServ model where packets arrive marked as belonging to a certain QoS class/DSCP, and when they are identified as excess, they should then be remarked to a different QoS Class (DSCP value) indicated in the 'Remark Value', as follows:

Remark Value (6 bits): indicates either drop (set to 0) or DSCP value to remark packets to when identified as excess

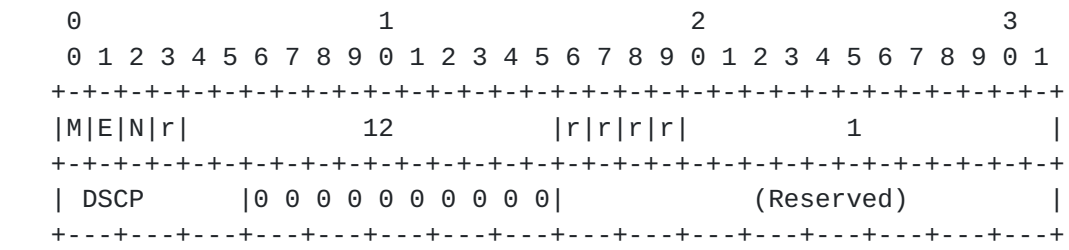
If 'no metering or policing is permitted' is signaled, the QNE should accept the excess treatment parameter set by the sender with special care so that excess traffic should not cause a problem. To request the Null Meter [RFC3290] is especially strong, and should be used with caution.

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.

5.2.12 <PHB Class> Parameter [RFC3140]

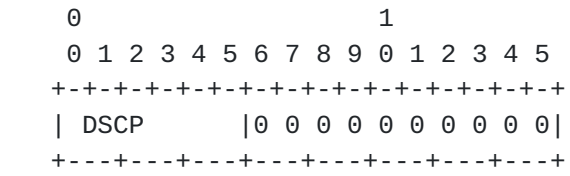
The coding for the <PHB Class> parameter is as follows:



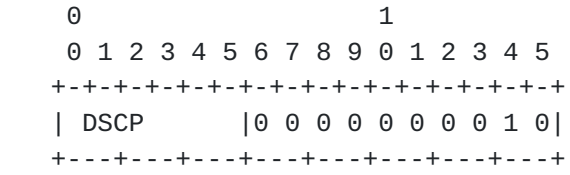
As prescribed in RFC 3140, the encoding for a single PHB is the recommended DSCP value for that PHB, left-justified in the 16 bit field, with bits 6 through 15 set to zero.

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.)

Single PHB:

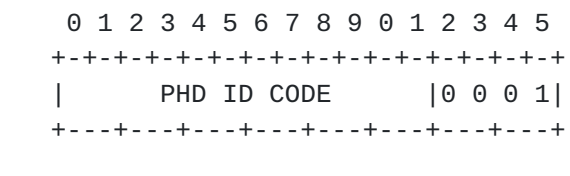


Set of PHBs:

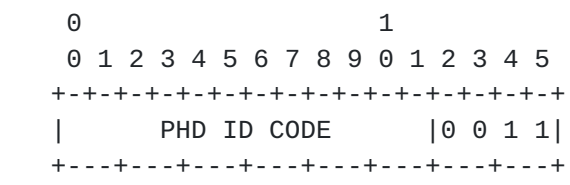


PHBs not defined by standards action, i.e., experimental or local use PHBs as allowed by [RFC2474]. In this case an arbitrary 12 bit PHB identification code, assigned by the IANA, is placed left-justified in the 16 bit field. Bit 15 is set to 1, and bit 14 is zero for a single PHB or 1 for a set of PHBs.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

Single non-standard PHB (experimental or local):



Set of non-standard PHBs (experimental or local):



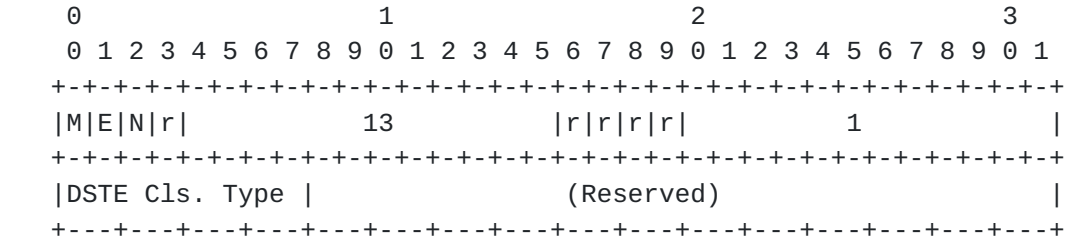
Bits 12 and 13 are reserved either for expansion of the PHB identification code, or for other use, at some point in the future.

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.

5.2.13 <DSTE Class Type> Parameter [RFC4124]

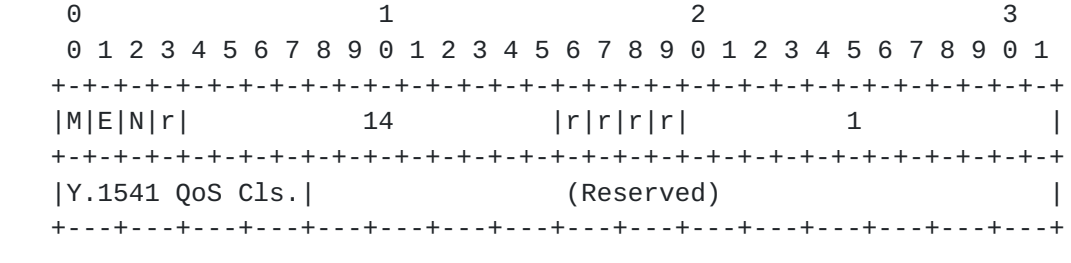
The coding for the <DSTE Class Type> parameter is as follows:



DSTE Class Type: Indicates the DSTE class type. Values currently allowed are 0, 1, 2, 3, 4, 5, 6, 7.

5.2.14 <Y.1541 QoS Class> Parameter [Y.1541]

The coding for the <Y.1541 QoS Class> parameter is as follows:



Y.1541 QoS Class: Indicates the Y.1541 QoS Class. Values currently allowed are 0, 1, 2, 3, 4, 5, 6, 7.

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.

6. Security Considerations

QSPEC security is directly tied to QoS NSLP security, and the QoS NSLP document [QoS-SIG] has a very detailed security discussion in Section 7. All the considerations detailed in Section 7 of [QoS-SIG] apply to QSPEC.

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)

7. 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 5:
Object Type = 0: QoS Desired
= 1: QoS Available
= 2: QoS Reserved
= 3: 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.

QSPEC Type (4 bits):
The following value is allocated by this specification:
0: Default
The allocation policies for further values are as follows:
1-63: Specification Required
64-255: Reserved

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 4.3:
Message Sequence 0:
Semantic: QSPEC Procedure = Two-Way Transaction

(see Section 4.3.1)
Message Sequence 1:
Semantic: QSPEC Procedure = Three-Way Transaction
(see Section 4.3.2)
Message Sequence 2:
Semantic: QSPEC Procedure = Resource Queries (see Section 4.3.3)
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 4.3:
Message Sequence: 0
Object Combination: 1, 2, 3
Semantic: see Table 1 in Section 4.3.1
Message Sequence: 1
Object Combination: 1, 2, 3
Semantic: see Table 2 in Section 4.3.2
Message Sequence: 2
Object Combination: 1, 2, 3
Semantic: see Table 3 in Section 4.3.3
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 Section 5.2:
Parameter ID 1: <TMOD-1>
2: <TMOD-2>
3: <Path Latency>
4: <Path Jitter>
5: <Path PLR>
6: <Path PER>
7: <Slack Term>
8: <Preemption Priority> & <Defending Priority>
9: <Admission Priority>
10: <RPH Priority>

```
                11: <Excess Treatment>
                12: <PHB Class>
                13: <DSTE Class Type>
                14: <Y.1541 QoS Class>
```

The allocation policies for further values are as follows:
15-255: Specification Required
256-1024: Private/Experimental Use
1025-4095: Reserved

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

Admission Priority Parameter (8 bits):
The following values are allocated by this specification:
0-2: assigned as specified in Section 6.2.9:
Admission Priority 0: best-effort priority flow
1: normal priority flow
2: high priority flow
The allocation policies for further values are as follows:
3-63: Standards Action
64-255: 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]), and an extension to this registry is created in [EMERGENCY-RSVP], which will also be used for the QSPEC RPH parameter. In the extended registry, "Namespace Numerical Values" are assigned by IANA to RPH Namespaces and "Priority Numerical Values" are assigned to the RPH Priority.

Excess Treatment Parameter (8 bits):

The following values are allocated by this specification:
0-3: assigned as specified in Section 5.2.11:
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 Parameter (8 bits):
The following values are allocated by this specification:
0-7: assigned as specified in Section 5.2.13:
DSTE Class Type 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 Parameter (8 bits):
The following values are allocated by this specification:
0-7: assigned as specified in Section 5.2.14:

Y.1541 QoS Class 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

8. 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, Martin Stiemerling, An Nguyen, Tom Phelan, James Polk, Alexander Sayenko, John Rosenberg, Bernd Schloer, Hannes Tschofenig, and Sven van den Bosch for their very helpful suggestions.

9. Contributors

This document is the result of the NSIS Working Group effort. In addition to the authors/editors listed in Section 12, the following people contributed to the document:

Roland Bless
Institute of Telematics, Universitaet Karlsruhe (TH)
Zirkel 2
Karlsruhe 76131
Germany
Phone: +49 721 608 6413
Email: bless@tm.uka.de
URI: http://www.tm.uka.de/~bless

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@research.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

10. 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).
[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.
[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.
[RFC3140] Black, D., et al., "Per Hop Behavior Identification Codes," June 2001.
[RFC3181] Herzog, S., "Signaled Preemption Priority Policy Element," RFC 3181, October 2001.
[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.
[RFC4506] Eisler, M., "XDR: External Data Representation Standard," RFC 4506, May 2006.
[Y.1541] ITU-T Recommendation Y.1541, "Network Performance Objectives for IP-Based Services," February 2006.
[Y.2171] ITU-T Recommendation Y.2171, "Admission Control Priority Levels in Next Generation Networks," September 2006.

11. Informative References

[DQOS] Cablelabs, "PacketCable Dynamic Quality of Service Specification," CableLabs Specification PKT-SP-DQOS-I12-050812, August 2005.
[EMERGENCY-RSVP] Le Faucheur, F., et. al., "Resource ReSerVation Protocol (RSVP) Extensions for Emergency Services," work in

progress.
[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.
[DSCP-REGISTRY] http://www.iana.org/assignments/dscp-registry
[NETWORK-BYTE-ORDER] Wikipedia, "Endianness," http://en.wikipedia.org/wiki/Endianness.
[PHBID-CODES-REGISTRY] http://www.iana.org/assignments/phbid-codes
[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 2434, 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.
[RFC2475] Blake, S., et al., "An Architecture for Differentiated Services", RFC 2475, December 1998.
[RFC2597] Heinanen, J., et al., "Assured Forwarding PHB Group," RFC 2597, June 1999.
[RFC2697] Heinanen, J., et al., "A Single Rate Three Color Marker," RFC 2697, September 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.
[RFC3290] Bernet, Y., et al., "An Informal Management Model for Diffserv Routers," RFC 3290, May 2002.
[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
[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.

12. Authors' Addresses

Gerald Ash (Editor)
AT&T
Email: gash5107@yahoo.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)
deZem GmbH
Knesebeckstr. 86/87
10623 Berlin
Germany
Email: cornelia.kappler@googlemail.com

David R. Oran (Editor)
Cisco Systems, Inc.
7 Ladyslipper Lane
Acton, MA 01720, USA
Email: oran@cisco.com

Appendix A. 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. TMOD). 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 B. Change History & Open Issues

This appendix should be removed by the RFC editor before publication.

B.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 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 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) 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 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 parameter coding from QSPEC parameter coding
- <Traffic>, <QoS Class>, and <Priority> QSPEC 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> from a variable length to a fixed length parameter

Version -13:

- notion of QOSMs played down
  - o language e.g. 'QNSLP/QSPEC can signal for different QOSMs across multiple domains' replaced by notion that 'QNSLP/QSPEC allows QNEs on the path to implement different data plane QoS mechanisms that meet QSPEC constraints'
  - o a QOSM describes common capabilities among QNEs to act consistently when requested to admit traffic & in treating admitted traffic
  - o a QOSM ID need not be defined or signaled
  - o a QNE need not support any particular QOSM although a QNI normally includes a QSPEC corresponding to a particular QOSM
- a 'QOSM specification'
  - o still provides a rigorous specification of a QOSM & what it does
  - o documents how a QNE interprets & treats various elements in QSPEC
  - o can define additional QSPEC parameters
- updated QOSM definition:
  a method to achieve QoS for a traffic flow, e.g., IntServ Controlled Load; specifies what sub-set of QSPEC QoS constraints & traffic handling directives a QNE implementing that QOSM is capable of supporting & how resources will be managed by the RMF
- QSPEC1/QSPEC2 semantics replaced with following semantics:
  - o source traffic description (mandatory to include by QNI & mandatory to interpret by downstream QNEs)
    - > specified by traffic model (TMOD-1) parameter consisting of rate (r), bucket size (b), peak rate (p), minimum policed unit (m) (mathematically complete way to describe traffic source)
    - > bandwidth only set r=p; b/m to large values (separate bandwidth parameter not needed)
    - > TMOD-2 (optional to include)
  - o constraints (optional to include):
    - > Path Latency
    - > Path Jitter
    - > Path PLR
    - > Path PER
    - > Slack Term
    - > Priority (Preemption, Defending, Admission, RPH Priority)
  - o handling directives (optional to include):
    - > Excess Treatment
  - o traffic classifiers (optional to include):
    - > PHB Class (PHB class set by downstream QNE to tell QNI how to mark traffic to ensure treatment committed by admission control)

> DSTE Class Type
> Y.1541 QoS class

o eliminated:
> Bandwidth

> Ctot, Dtot, Csum, Dsum
- concept of remapping QSPEC parameters eliminated
- redefine 'interpret' a QSPEC parameter to mean 'must conform to RFCs defining parameter & procedures (formerly called 'strictly interpret')
- concept of local QSPECs retained
  o allows simpler control plane in a local domain
  o edge nodes
    > must interpret initiator QSPEC parameters
    > can either initiate parallel session with local QSPEC or define a local QSPEC with encapsulated initiator QSPEC
  o local QSPEC interpreted by local domain QNEs
  o local QSPEC must be consistent with initiator QSPEC
    > e.g., RMD can initiate a local QSPEC that contains TMOD = bandwidth (sets r=p, b/m to large)
- QSPEC flags modified as follows:
  o QNI sets M flag for each QSPEC parameter it populates that must be interpreted or reservation fails
  o if M flag set
    > downstream QNE MUST interpret parameter or reservation fails
    > if QNE does not support parameter it sets N flag & rejects reservation
    > if QNE supports parameter but cannot meet parameter, it sets E flag & rejects reservation
  o if M flag not set
    > downstream QNE SHOULD interpret parameter
    > if QNE does not support parameter it sets the N flag & optionally accepts or rejects reservation
    > if QNE supports parameter but cannot meet parameter, it sets E flag & optionally accepts or rejects reservation
  o R (remapped parameter) flag & Q (non QOSM) flag eliminated

Version -14:

- Section 4.3.3 added text that QOSM specifications SHOULD NOT define new RMF functions
- Section 5.1 added text that both mechanisms can be used simultaneously: a) tunneling a QSPEC through a local domain and b) defining a local QSPEC and encapsulating the initiator QSPEC
- Section 4.1 added text that signaling functionality is only defined by the QoS NSLP document
- Section 4.1 added text that QOSMs are free to extend QSPECs by adding parameters but are not permitted to reinterpret or redefine the standard QSPEC parameters specified in this document

Version -15:

- editorial revisions made to Sections 4.1, 4.3.3, 5.3.1, 5.3.2, and

5.3.3 according to agreements on NSIS discussion list archive.

Version -16:

- QSPEC Types: additional QSPEC Types can be defined per IANA Considerations Section (already in place); QSPEC Type = 0 is default
- Initiator/Local QSPEC bit added
- various editorial fixes: DSCP parameter encoding; various edits carry over from QSPEC-1 parameter removal; QSPEC version number edits & additional error code

Version -17:
- QSPEC Header format: added Length field

Version -18:
- clarified handling of Traffic Handling Directives in QoS Available in Sec. 4.3.3
- classified Priority Parameters as Traffic Handling Directives (Previously and erroneously were classified as Constraint Parameters)
- added units to TMOD parameter in 6.2.1
- fixed error in possible object combination for Resource Queries in Sec 6.1
- streamlined usage of QSPEC Type and added terminology

Version -19:
- changes made per NSIS chair review (as specified on list)

B.2 Open Issues

None.

Intellectual Property Statement

this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Full Copyright Statement

Disclaimer of Validity