                              QSPEC Template
        for the Quality-of-Service NSIS Signaling Layer Protocol (NSLP)

Abstract

   The Quality-of-Service (QoS) NSIS signaling layer protocol (NSLP) 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 reused in several QOSMs and thereby aim to ensure the
   extensibility and interoperability of QoS NSLP. While the base
   protocol is QOSM-agnostic, the parameters that can be carried in the
   QSPEC object are possibly closely coupled to specific models. 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.

Status of This Memo

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc5975.

Table of Contents

1. Introduction

   The QoS NSIS signaling layer protocol (NSLP) [RFC5974] 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). [RFC5974] describes the
   signaling protocol, while this document describes the RMF-related
   information carried in the QSPEC (QoS Specification) object carried
   in QoS NSLP messages.

   [RFC5974] 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 be familiar with [RFC5974] 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
   [CL-QOSM], [RFC5976], and [RFC5977].

QSPEC parameters include, for example:

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

While the base protocol is QOSM-agnostic, the parameters that can be
carried in the QSPEC object are possibly closely coupled to specific
models.

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 [RFC5974]. 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, enable simpler processing by QNEs within the local
domain.

In Section 3.4, an example of QSPEC processing is provided.

1.1. 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].

2. Terminology

   Initiator QSPEC: The Initiator QSPEC is included in 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: QSPEC object that, together with a description of QoS
   Desired or QoS Available, 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. Note that the term "minimum" is used
   generically, since for some parameters, such as loss rate and
   latency, what is specified is the maximum acceptable value.

   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 the subset of QSPEC QoS
   constraints and traffic handling directives that a QNE implementing
   that QOSM is capable of supporting and how resources will be managed
   by the RMF.

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

   QSPEC: the object of QoS NSLP that contains 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 the 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 [RFC5974] 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
[RFC5977], and QoS signaling for Y.1541 QoS classes [RFC5976]. 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 and the
information carried in QSPEC objects. Under some circumstances
(e.g., aggregation), they may 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 [RFC5974]. New QoS NSLP message
processing rules can only be defined in extensions to QoS NSLP. If a
QOSM specification defines triggers that deviate from existing QoS
NSLP processing rules, the fallback for QNEs not supporting that QOSM
are the 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 QOSMs 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 SHOULD 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 that the QOSM
  specification defines
- a definition of what happens in case of preemption if the default
  QNI behavior (teardown preempted reservation) is not followed (see
  Section 4.3.5)

A QOSM specification MAY include the following:

- definitions of additional QOSM-specific error codes, as discussed
  in Section 4.2.3
- the QoS-NSLP options a QOSM wants to use, when several options are
  available for a QOSM (e.g., Local QSPEC to either a) hide the
  Initiator QSPEC within a local domain message, or b) encapsulate
  the 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 QSPEC. This includes traffic description parameters, constraint
parameters, and traffic handling directives. QOSMs are not
permitted, however, to reinterpret or redefine the QSPEC parameters
specified in this document. Note that signaling functionality is
only defined by the QoS NSLP document [RFC5974] 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 the traffic model parameter (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 the current version is
   deprecated or deleted (as required by a specification). 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 4 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, QoS Reserved and Minimum
   QoS MUST all be supported by QNEs and MAY appear in any QSPEC object
   carried in any QoS NSLP message (RESERVE, QUERY, RESPONSE, NOTIFY).
   See [RFC5974] for descriptions of the QoS NSLP RESERVE, QUERY,
   RESPONSE, and NOTIFY messages.

```
   +--------------------------------------+
   |            QSPEC Objects             |
   +--------------------------------------+

   _____ _____/
            V
   +----------+----------+---------+-------+
   |QoS Desir.|QoS Avail.|QoS Rsrv.|Min QoS|
   +----------+----------+---------+-------+

   \____ ____/\___ _____/\___ ____/\__ ___/
      V         V          V        V

   +------------+...      +------------+...
   |QSPEC Para. 1|        |QSPEC Para. n|
   +------------+...      +------------+...
```
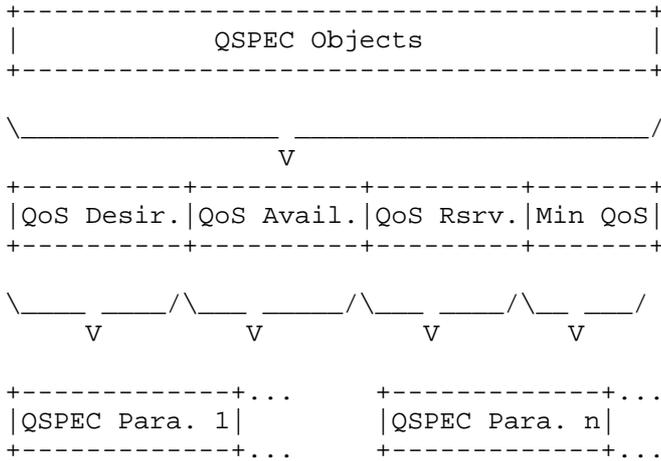
         Figure 1: Structure of the QSPEC

   Use of the 4 QSPEC objects (QoS Desired, QoS Available, QoS Reserved,
   and Minimum QoS) is described in Section 4.3 for 3 message sequences
   and 7 object combinations.

   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 and sometimes included in the
   QUERY message (see Section 4.3 for details).

   As described in Section 4.3, the QoS Available object may travel in a
   RESERVE message, RESPONSE Message, or QUERY message and may collect
   information on the resources currently available on the path. In
   this case, QoS Available is a read-write object, which means the
   QSPEC parameters contained in QoS Available may be updated, but they
   cannot be deleted. As such, 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. In one other instance described in
Section 4.3.2 (Case 3), QoS Available is sent by the QNI in a RESERVE
message as a read-only QSPEC object (see Section 4.3.2 for details).

The QoS Reserved object reflects the resources that are being
reserved. It is a read-only object and is always included in a
RESPONSE message if QoS Desired is included in the RESERVE message
(see Section 4.3 for details).

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.
Note that the term "minimum" is used generically, since for some
parameters, such as loss rate and latency, what is specified is the
maximum acceptable value. It is a read-only object, and may be
included in a RESERVE message, RESPONSE message, or QUERY message
(see Section 4.3 for details). 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 signals back to the QNI the
value of QoS Available attained in the end, because the reservation
may need to be adapted accordingly (see Section 4.3 for details).

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 [RFC5976]
and [RFC5977].

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 5 sub-parameters [RFC2212]:

o rate (r) specified in octets per second
o bucket size (b) specified in octets
o peak rate (p) specified in octets per second
o minimum policed unit (m) specified in octets
o maximum packet size (MPS) specified in octets

The TMOD parameter takes the form of a token bucket of rate (r) and
bucket size (b), plus a peak rate (p), minimum policed unit (m), and
maximum packet size (MPS).

Both b and r MUST be positive. The rate, r, is measured in octets of
IP packets per second, and can range from 1 octet per second to as
large as 40 teraoctets per second. The bucket depth, b, is also
measured in octets and can range from 1 octet to 250 gigaoctets. The
peak rate, p, is measured in octets of IP packets per second and has
the same range and suggested representation as the bucket rate.

The peak rate is the maximum rate at which the source and any
reshaping (defined below) may inject bursts of traffic into the
network. More precisely, it is a requirement that for all time
periods the amount of data sent cannot exceed MPS+pT, where MPS is

the maximum packet size and T is the length of the time period.
Furthermore, p MUST be greater than or equal to the token bucket
rate, r. If the peak rate is unknown or unspecified, then p MUST be
set to infinity.

The minimum policed unit, m, is an integer measured in octets. All
IP packets less than size m will be counted, when policed and tested
for conformance to the TMOD, as being of size m.

The maximum packet size, MPS, is the biggest packet that will conform
to the traffic specification; it is also measured in octets. The
flow MUST be rejected if the requested maximum packet size is larger
than the MTU of the link. Both m and MPS MUST be positive, and m
MUST be less than or equal to MPS.

Policing compares arriving traffic against the TMOD parameters at the
edge of the network. Traffic is policed to ensure it conforms to the
token bucket. Reshaping attempts to restore the (possibly distorted)
traffic's shape to conform to the TMOD parameters, and traffic that
is in violation of the TMOD is discovered because the reshaping fails
and the reshaping buffer overflows.

The token bucket and peak rate parameters require that traffic MUST
obey the rule that over all time periods, the amount of data sent
cannot exceed MPS+min[pT, rT+b-MPS], where r and b are the token
bucket parameters, MPS is the maximum packet size, and T is the
length of the time period (note that when p is infinite, this reduces
to the standard token bucket requirement). For the purposes of this
accounting, links MUST count packets that are smaller than the
minimum policing unit as being of size m. Packets that arrive at an
element and cause a violation of the MPS + min[pT, rT+b-MPS] bound
are considered non-conformant.

All 5 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, and m = large. As another example, if TMOD is
set for TCP traffic, then set r = average rate, b = large, and p =
large.

When the 5 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, p, and MPS, 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 parameter (<TMOD-2>) is specified as could be needed
to support some Diffserv applications. For example, it is typically
assumed that Diffserv Expedited Forwarding (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 Assured Forwarding (AF) traffic, two sets of token bucket
parameters are needed -- one for the average traffic and one 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, packet
   loss ratio, and path packet error ratio, 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, measured in microseconds, added
   by each QNE. This delay results from the combination of link
   propagation delay, packet processing, and queuing. Each QNE MUST add
   the propagation delay of its outgoing link, if this link exists.

Furthermore, the QNI SHOULD 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 that 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, measured in microseconds, 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 [RFC3393]
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 that may be present. Each QNE MUST add the
jitter of its outgoing link, if this link exists. Furthermore, the
QNI SHOULD 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 is the unit-less ratio of total lost IP
packets to total transmitted IP packets. <Path PLR> 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.

Packet error ratio [Y.1540, Y.1541] is the unit-less ratio of total
errored IP packet outcomes to the total of successful IP packet
transfer outcomes plus errored IP packet outcomes in a population of

interest, with a resolution of at least 10^-9. If lesser resolution
is available in a value, the unused digits MUST be set to zero. Note
that the number of errored packets observed is directly related to
the confidence in the result. 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 SHOULD 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 it 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 [RFC5974]. Note,
however, that new QoS NSLP message processing rules can only be
defined in 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 [RFC5976],
[RFC5977], and [CL-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,
Emergency Telecommunications Service (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 re-marked.

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 the treatment that is designated 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.,
[RFC5976], [RFC5977], 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.

```
+----------+       /-------\       /--------\       /--------\
| Laptop   |       | Home  |       | Cable  |       | Diffserv |
| Computer |-----| Network |-----| Network  |-----| Network  |----+
+----------+       | No QOSM |       |DQOS QOSM |       | RMD QOSM |    |
                   \-------/       \--------/       \--------/    |
                                                                  |
          +-------------------------------------------------+
          |
          |     /--------\       +----------+
          |     |   XG    |       | Handheld |
          +---| Wireless |-----| Device   |
                | XG QOSM |       +----------+
                \--------/
```
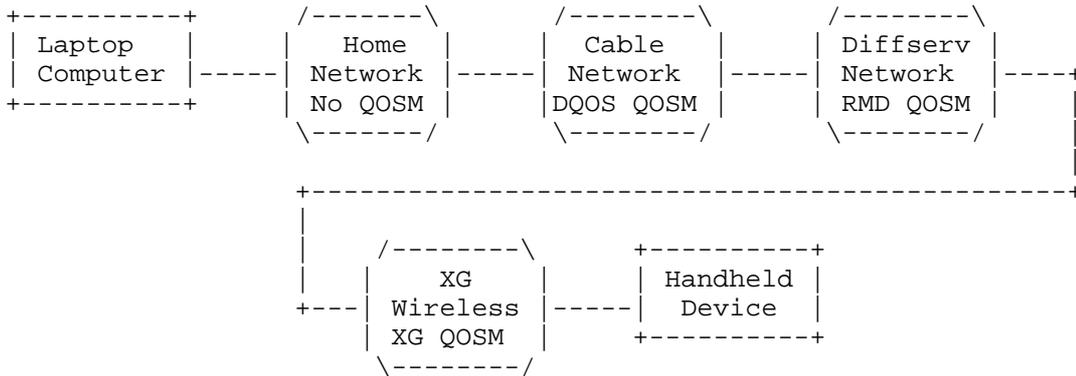
        Figure 2: Example Configuration of QoS-NSLP/QSPEC Operation

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
Resource Management in Diffserv QoS Model [RFC5977]. 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 the 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; they
are 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 QoS guarantees similar to IntServ Controlled Load service,
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 <PHB 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; they are described 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
     [RFC5974]). 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 inter-domain (per-flow)
     RESERVE, but the aggregate reservation is updated by the border QNE
     (or QNI) as need be. This is also how RMD works [RFC5977].

   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 [RFC5974] and in [RFC5977]. The ingress QNE to the
   RMD domain maps the TMOD parameters contained in the original
   Initiator QSPEC to 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, if the node can, at the egress to the RMD domain, it
   updates QoS Available on behalf of the entire RMD domain. If it
   cannot (since the M flag is not set for <Path Latency>), it raises
   the parameter-specific, Not Supported N 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 aborts the RESERVE and sends 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 [RFC5974].

For bit-level examples of QSPECs, see the documents specifying QOSMs:
[CL-QOSM], [RFC5976], and [RFC5977].

4. QSPEC Processing and 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 the 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 and 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) 'hide' 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 the 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).

```
+------------------------------+\
|   QSPEC Type, QSPEC Procedure | \
+------------------------------+ / Common QSPEC Header
|   Init./Local QSPEC bit=Local  |/
+==============================+\
| Local-QSPEC Parameter 1      | \
+------------------------------+  \
|            ....              |   Local-QSPEC Parameters
+------------------------------+  /
| Local-QSPEC Parameter n      | /
+------------------------------+/
| +----------------------------+ |
| | QSPEC Type, QSPEC Procedure| |
| +----------------------------+ |
| | Init./Local QSPEC bit=Init.| |
| +============================+ |
| |                            | | Encapsulated Initiator QSPEC
| |          ....              | |
| +----------------------------+ |
+------------------------------+
```
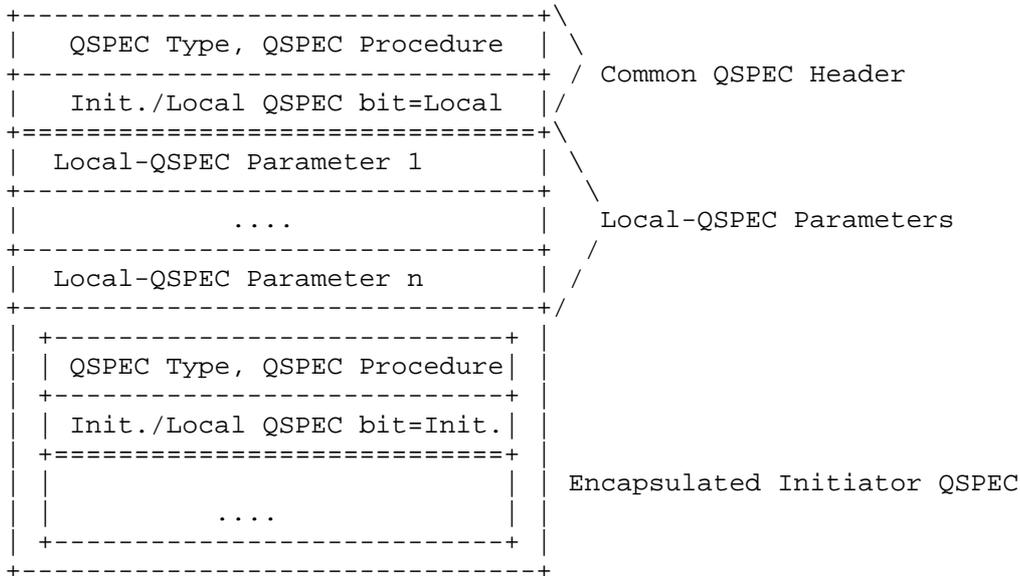
                  Figure 3: Defining a Local QSPEC

Here the QoS-NSLP only sees and passes one QSPEC up to the RMF.
Thus, the type of the QSPEC 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, and

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 to 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 used at the same time for the same
flow.

The support of Local QSPECs 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 a 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 <QoS Desired>, <Minimum QoS>, and <QoS
Available> objects to include objectives for the <Path Latency>,
<Path Jitter>, and <Path PER> 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 <QoS Available> 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.

```
|------|    |------|                                    |------|    |------|
| e2e  |<->| e2e  |<----------------------------------->| e2e  |<->| e2e  |
| QOSM |    | QOSM |                                     | QOSM |    | QOSM |
|      |    |------|    |-------|    |-------|    |------|    |      |
| NSLP |    | NSLP |<->| NSLP  |<->| NSLP  |<->| NSLP |    | NSLP |
|Y.1541|    |local |    |local  |    |local  |    |local |    |Y.1541|
| QOSM |    | QOSM |    | QOSM  |    | QOSM  |    | QOSM |    | QOSM |
|------|    |------|    |-------|    |-------|    |------|    |------|
-----------------------------------------------------------------------
|------|    |------|    |-------|    |-------|    |------|    |------|
| NTLP |<->| NTLP |<->| NTLP  |<->| NTLP  |<->| NTLP |<->| NTLP |
|------|    |------|    |-------|    |-------|    |------|    |------|
  QNI        QNE         QNE          QNE         QNE        QNR
 (End)  (Ingress Edge) (Interior)  (Interior) (Egress Edge)  (End)
```
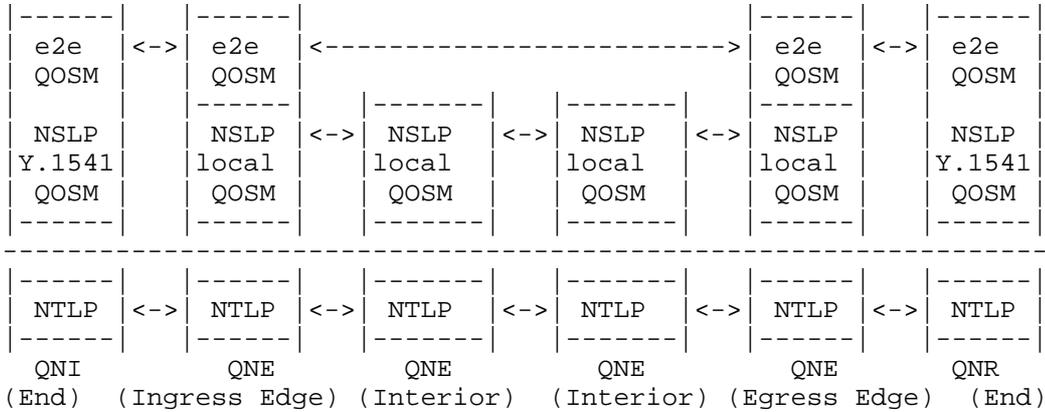
     Figure 4: Example of Initiator and Local Domain QOSM Operation

4.2. Reservation Success/Failure, QSPEC Error Codes, and 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. However, the reservation 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 provide details on the handling of
   unsuccessful reservations and reservations where some parameters
   could not be met, 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 [RFC5974], 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.

   Note that when there are routers along the path between the QNI and
   QNR where QoS cannot be provided, then the QoS-NSLP generic flag
   BREAK (B) is set. The BREAK flag is discussed in Section 3.3.5 of
   [RFC5974].

4.2.1. Reservation Failure and 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 the E flag set to indicate
   whether the parameter could or could not be satisfied. 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 down 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 indicate 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 [RFC5974], 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 ("QSPEC
   Procedures"). The RESPONSE message MAY omit the QSPEC in case of a
   successful reservation.

   The following guidelines are provided for setting the error codes in
   the INFO-SPEC, based on the codes provided in Section 5.1.3.6 of
   [RFC5974]:

   - NSLP error class 2 (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.

   - NSLP error class 4 (Transient Failure) / 0x07 (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 at each QNE 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.

   - NSLP error class 3 (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.

   - NSLP error class 3 (Protocol Error) / 0x0f (Incompatible QSPEC):
     A higher version QSPEC is signaled and not supported by the QNE.

   - NSLP error class 6 (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
     Section 4.3 ("QSPEC Procedures").

   - 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
     [RFC5974], 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 [RFC5974], 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 [RFC5974].

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. However, when the message (RESPONSE
in case of stateful QNEs; RESERVE in case of stateless QNEs)
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, even if there
is no direct match between the parameters in the local and
Initiator QSPECs, E flags could still 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 [RFC5974]. 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:

MESSAGE SEQUENCE	OBJECT COMBINATION	OBJECTS INCLUDED IN RESERVE MESSAGE	OBJECTS INCLUDED IN RESPONSE MESSAGE
0	0	QoS Desired	QoS Reserved
0	1	QoS Desired QoS Available	QoS Reserved QoS Available
0	2	QoS Desired QoS Available Minimum QoS	QoS Reserved QoS Available

Table 1: Message Sequence 0: Two-Way Transactions
        Defining Object Combinations 0, 1, and 2

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 that 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:

| MSG. | OBJ. | OBJECTS INCLUDED | OBJECTS INCLUDED | OBJECTS INCLUDED |
SEQ.	COM.	IN QUERY MESSAGE	IN RESERVE MESSAGE	IN RESPONSE MESSAGE
1	0	QoS Desired	QoS Desired	QoS Reserved
1	1	QoS Desired	QoS Desired	QoS Reserved
		(Minimum QoS)	QoS Available	QoS Available
			(Minimum QoS)	
1	2	QoS Desired	QoS Desired	QoS Reserved
		QoS Available	QoS Available	

Table 2: Message Sequence 1: Three-Way Transactions
        Defining Object Combinations 0, 1, and 2

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 QNI 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 be 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 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.

| MESSAGE | OBJECT | OBJECTS INCLUDED | OBJECTS INCLUDED |
SEQUENCE	COMBINATION	IN QUERY MESSAGE	IN RESPONSE MESSAGE
2	0	QoS Available	QoS Available

Table 3: Message Sequence 2: Resource Queries
Defining Object Combination 0

Note that the QoS Available object when traveling in the QUERY
message can be overwritten, whereas in the RESPONSE message it
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
    unidirectional 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 the
    QNE, and the QNE sends a NOTIFY message to the QNI, as described in
    [RFC5974]. The NOTIFY message carries an INFO-SPEC with the error
    code as described in [RFC5974]. 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 [RFC5977] and [RFC5976].

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

5.1. General QSPEC Formats

    The format of the QSPEC closely follows that used in GIST [RFC5971]
    and QoS NSLP [RFC5974]. Every object (and parameter) has the
    following general format:

o The overall format is Type-Length-Value (in that order).

o Some parts of the type field are set aside for control flags.

o 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.

o 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.

o 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.

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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Common QSPEC Header                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                       QSPEC Objects                         //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
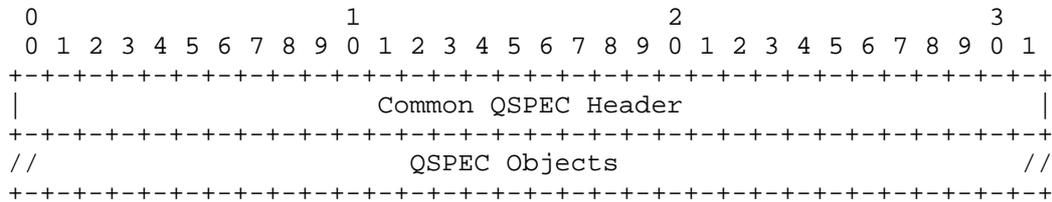
5.1.1. Common Header Format

   The Common QSPEC Header is a fixed 4-octet object containing the
   QSPEC Version, QSPEC Type, an identifier for the QSPEC Procedure (see
   Section 4.3), and an Initiator/Local QSPEC bit:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Vers.|I|QSPECType|r|r|  QSPEC Proc. |         Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
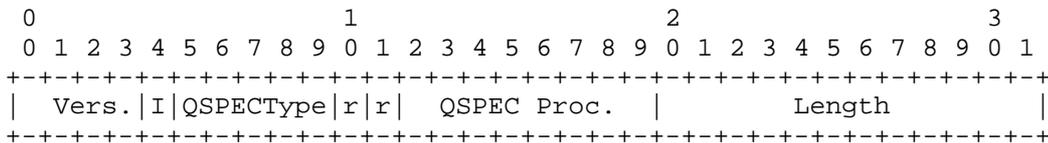
   Vers.: Identifies the QSPEC version number. QSPEC Version 0 is
          assigned by this specification in Section 7 (IANA
          Considerations).

   QSPEC Type: Identifies the particular type of QSPEC, e.g., a QSPEC
               Type corresponding to a particular QOSM. QSPEC Type 0
               (default) is assigned by this specification in Section 7
               (IANA Considerations).

   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:

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

               The Message Sequence field can attain the following
               values:

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

               The Object Combination field can take the values between
               1 and 3 indicated in the tables in Section 4.3:

               Message Sequence: 0
               Object Combination: 0, 1, 2
               Semantic: see Table 1 in Section 4.3.1

               Message Sequence: 1
               Object Combination: 0, 1, 2
               Semantic: see Table 2 in Section 4.3.2

               Message Sequence: 2
               Object Combination: 0
               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 (in 32-bit words) 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:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|r|r|r|        Object Type      |r|r|r|r|          Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
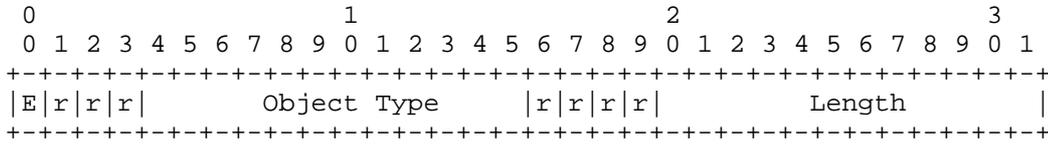
E Flag: Set if an error occurs on object level

Object Type = 0: QoS Desired (parameters cannot be overwritten)
            = 1: QoS Available (parameters may be overwritten; see
                 Section 3.2)
            = 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 encoded in the same
way in TLV format using a similar parameter header:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|E|N|r|      Parameter ID       |r|r|r|r|          Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
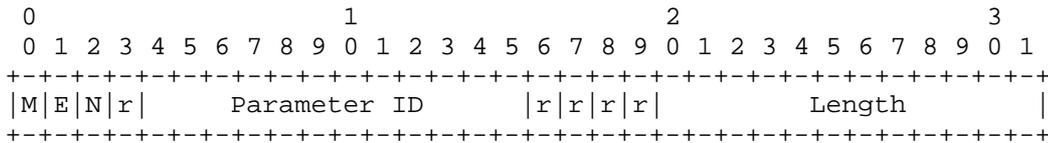
M Flag: When set, indicates the subsequent parameter MUST be
        interpreted. Otherwise, the parameter can be ignored if not
        understood.

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 consecutively to each QSPEC parameter.
              Parameter IDs are assigned to each QSPEC parameter
              defined in this document in Sections 5.2 and 7 (IANA
              Considerations).

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 5 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 [RFC5977])
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 (Per Hop Reservation) container parameter specified in [RFC5977].

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

The <TMOD-1> parameter consists of the <r>, <b>, <p>, <m>, and <MPS>
sub-parameters [RFC2212], 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:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|E|0|r|           1           |r|r|r|r|           5           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   TMOD Rate-1 (r) (32-bit IEEE floating point number)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   TMOD Size-1 (b) (32-bit IEEE floating point number)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Peak Data Rate-1 (p) (32-bit IEEE floating point number)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Minimum Policed Unit-1 (m) (32-bit unsigned integer)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Maximum Packet Size-1 (MPS) (32-bit unsigned integer)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The <TMOD-1> parameters are represented by three floating point
numbers in single-precision IEEE floating point format [IEEE754]
followed by two 32-bit integers in network octet 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), the first unsigned integer is the minimum policed unit (m), and
   the second unsigned integer is the maximum packet size (MPS). The
   values of r and p are measured in octets per second; b, m, and MPS
   are measured in octets. 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

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

   The coding for the <TMOD-2> parameter is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|E|N|r|         2           |r|r|r|r|         5               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   TMOD Rate-2 (r) (32-bit IEEE floating point number)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   TMOD Size-2 (b) (32-bit IEEE floating point number)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Peak Data Rate-2 (p) (32-bit IEEE floating point number)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Minimum Policed Unit-2 (m) (32-bit unsigned integer)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Maximum Packet Size-2 (MPS) (32-bit unsigned integer)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   The <TMOD-2> parameters are represented by three floating point
   numbers in single-precision IEEE floating point format [IEEE754]
   followed by two 32-bit integers in network octet 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), the first unsigned integer is the minimum policed unit (m), and
   the second unsigned integer is the maximum packet size (MPS). The
   values of r and p are measured in octets per second; b, m, and MPS
   are measured in octets. 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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|E|N|r|         3             |r|r|r|r|         1             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Path Latency (32-bit unsigned integer)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
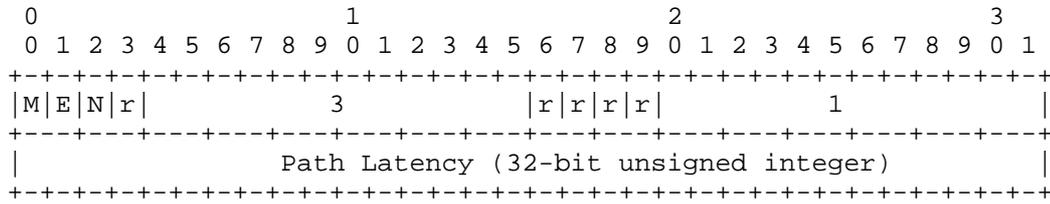
The Path Latency [RFC2215] is a single 32-bit unsigned integer in
network octet order. The intention of the Path Latency parameter is
the same as the Minimal Path Latency parameter defined in Section 3.4
of [RFC2215]. The purpose of this parameter is to provide a baseline
minimum path latency for use with services that provide estimates or
bounds on additional path delay, such as in [RFC2212]. Together with
the queuing delay bound offered by [RFC2212] and similar services,
this parameter gives the application knowledge of both the minimum
and maximum packet delivery delay.

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 add 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 cumulative 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 N 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

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |M|E|N|r|        4          |r|r|r|r|            4              |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
   |     Path Jitter STAT1(variance) (32-bit unsigned integer)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Path Jitter STAT2(99.9%-ile) (32-bit unsigned integer)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Path Jitter STAT3(minimum Latency) (32-bit unsigned integer)|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Path Jitter STAT4(Reserved)    (32-bit unsigned integer) |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
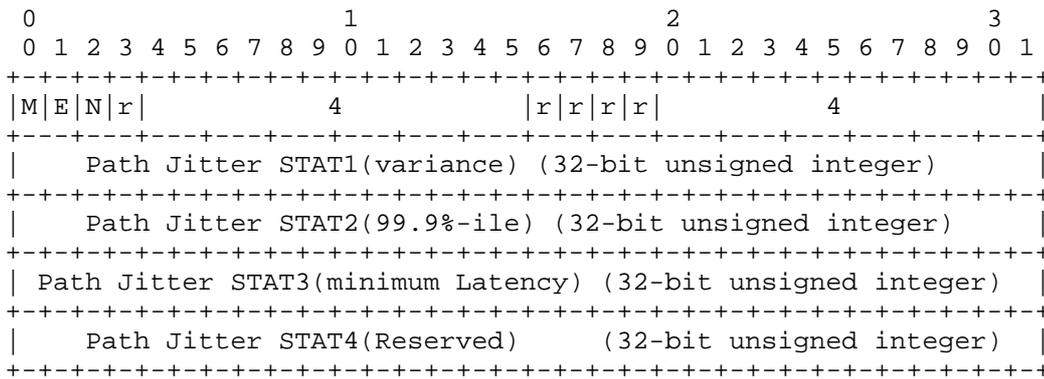
   The Path Jitter is a set of four 32-bit unsigned integers in network
   octet order [RFC3393, Y.1540, Y.1541]. As noted in Section 3.3.2,
   the Path Jitter parameter is called "IP Delay Variation" in
   [RFC3393]. The Path Jitter parameter is the combination of four
   statistics describing the Jitter 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
   add 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 cumulative 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]. Development continues on methods to combine
   jitter values to estimate the value of the complete path, and
   additional statistics may be needed to support new methods (the
   methods are standardized in [RFC5481] and [COMPOSITION]).

5.2.5. <Path PLR> Parameter

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|E|N|r|         5             |r|r|r|r|         1             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Path Packet Loss Ratio (32-bit floating point)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
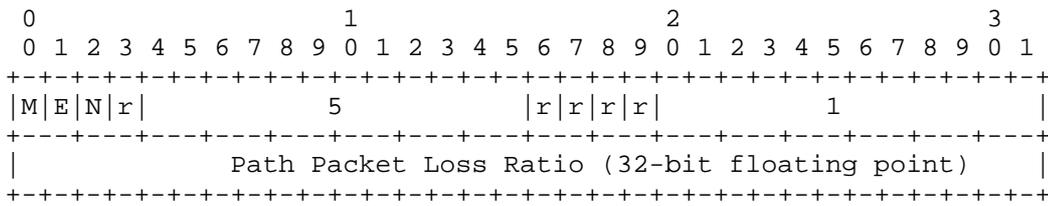
   The Path PLR is a single 32-bit single precision IEEE floating point
   number in network octet order [Y.1541]. As defined in [Y.1540], Path
   PLR is the ratio of total lost IP packets to total transmitted IP
   packets. An evaluation interval of 1 minute is suggested in
   [Y.1541], in which the number of losses observed is directly related
   to the confidence in the result. 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 adds its local PLR value (up to a maximum of 10^{-2}) to
   the total Path PLR value (up to a maximum of 10^{-1}) , where the
   acceptability of the total Path PLR value added across all QNEs is
   determined based on the QOSM being used. The maximum limit of 10^{-2}
   on a QNE's local PLR value and the maximum limit (clamp value) of
   10^{-1} on the accumulated end-to-end Path PLR value are used to
   preserve the accuracy of the simple additive accumulation function
   specified and to avoid more complex accumulation functions.
   Furthermore, if these maximums are exceeded, then the path would
   likely not meet the QoS objectives. If the sum of the different
   elements' values exceeds 10^{-1}, the end-to-end cumulative 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
   that 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

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |M|E|N|r|         6         |r|r|r|r|         1                 |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |             Path Packet Error Ratio (32-bit floating point)  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
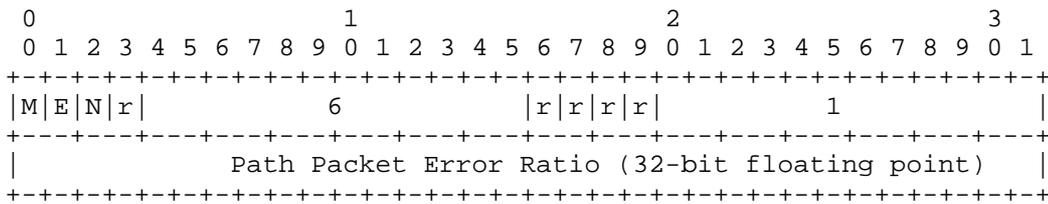
The Path PER is a single 32-bit single precision IEEE floating point
number in network octet order [Y.1541]. As defined in [Y.1540], Path
PER is the ratio of total errored IP packets to the total of
successful IP Packets plus errored IP packets, in which the number of
errored packets observed is directly related to the confidence in the
result. 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 adds its
local PER value (up to a maximum of 10^{-2}) to the total Path PER
value (up to a maximum of 10^{-1}) , where the acceptability of the
total Path PER value added across all QNEs is determined based on the
QOSM being used. The maximum limit of 10^{-2} on a QNE's local PER
value and the maximum limit (clamp value) of 10^{-1} on the accumulated
end-to-end Path PER value are used to preserve the accuracy of the
simple additive accumulation function specified and to avoid more
complex accumulation functions. Furthermore, if these maximums are
exceeded, then the path would likely not meet the QoS objectives. If
the sum of the different elements' values exceeds 10^{-1}, the end-to-
end cumulative PER SHOULD be reported as indeterminate. A QNE that
cannot accurately predict the PER of packets it is processing MUST
raise the Not Supported N flag and either leave the value of Path PER
as is, or add its best estimate of its lower bound. A raised Not
Supported N 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 that 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

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |M|E|N|r|         7             |r|r|r|r|         1             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Slack Term (S)  (32-bit unsigned integer)            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
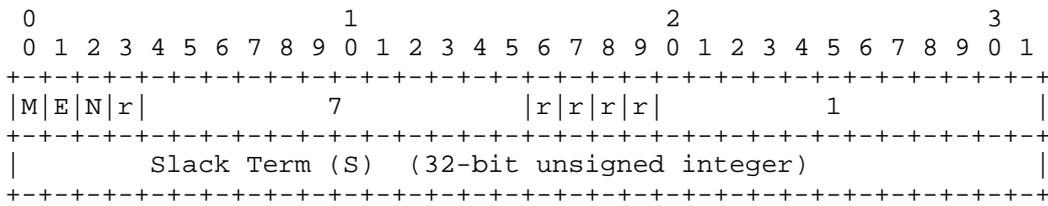
   Slack term S MUST be nonnegative and is measured in microseconds
   [RFC2212]. The Slack term, S, is represented as a 32-bit unsigned
   integer. Its value can range from 0 to $(2^{32})-1$ microseconds.

5.2.8. <Preemption Priority> and <Defending Priority> Parameters

   The coding for the <Preemption Priority> and <Defending Priority>
   sub-parameters is as follows [RFC3181]:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |M|E|N|r|         8             |r|r|r|r|         1             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Preemption Priority       |     Defending Priority        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
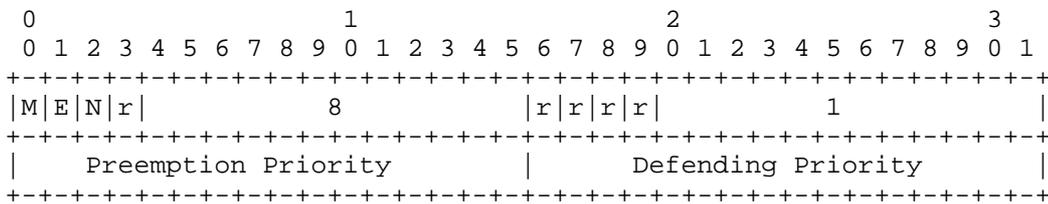
   Preemption Priority: The priority of the new flow compared with the
      defending priority of previously admitted flows. Higher values
      represent higher priority.

   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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|E|N|r|         9             |r|r|r|r|          1            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Y.2171 Adm Pri.|Admis. Priority|          (Reserved)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
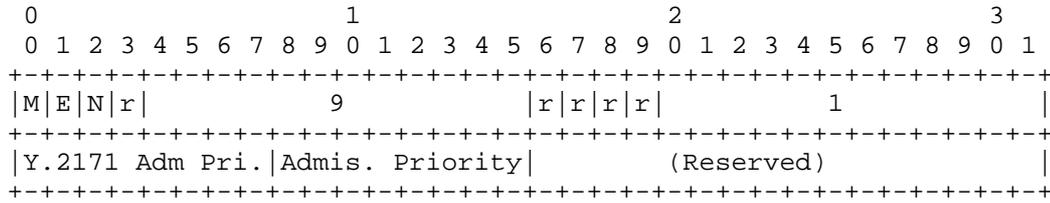
   Two fields are provided for the <Admission Priority> parameter and
   are populated according to the following rules.

   <Y.2171 Admission Priority> values are globally significant on an
   end-to-end basis. 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:

   <Y.2171 Admission Priority>:

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

   If the QNI signals <Y.2171 Admission Priority>, it populates both the
   <Y.2171 Admission Priority> and <Admission Priority> fields with the
   same value. Downstream QNEs MUST NOT change the value in the <Y.2171
   Admission Priority> field so that end-to-end consistency is
   maintained and MUST treat the flow priority according to the value
   populated. A QNE in a local domain MAY reset a different value of
   <Admission Priority> in a Local QSPEC, but (as specified in Section
   4.1) the Local QSPEC MUST be consistent with the Initiator QSPEC.
   That is, the local domain MUST specify an <Admission Priority> in the
   Local QSPEC that is functionally equivalent to the <Y.2171 Admission
   Priority> specified by the QNI in the Initiator QSPEC.

   If the QNI signals admission priority according to [EMERGENCY-RSVP],
   it populates a locally significant value in the <Admission Priority>
   field and places all ones in the <Y.2171 Admission Priority> field.
   In this case, the functional significance of the <Admission Priority>
   value is specified by the local network administrator. Higher values
   indicate higher priority. Downstream QNEs and RSVP nodes MAY reset
   the <Admission Priority> value according to the local rules specified
   by the local network administrator, but MUST NOT reset the value of
   the <Y.2171 Admission Priority> field.

   A reservation without an <Y.2171 Admission Priority> parameter MUST
   be treated as a reservation with an <Y.2171 Admission Priority> = 1.

5.2.10. <RPH Priority> Parameter

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|E|N|r|            10         |r|r|r|r|            1          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         RPH Namespace         | RPH Priority  |  (Reserved)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
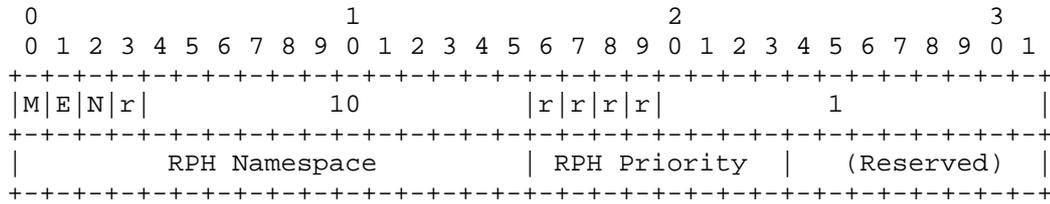
   [RFC4412] defines a resource priority header (RPH) with parameters
   "RPH Namespace" and "RPH Priority", and if populated is applicable
   only to flows with high admission priority. A registry is created in
   [RFC4412] and extended in [EMERG-RSVP] for IANA to assign the RPH
   priority parameter. 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>.)

Functionality similar to enhanced Multi-Level Precedence and
Preemption service (eMLPP; 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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|E|N|r|          11          |r|r|r|r|          1             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Excess Trtmnt |Re-mark Val|           Reserved               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
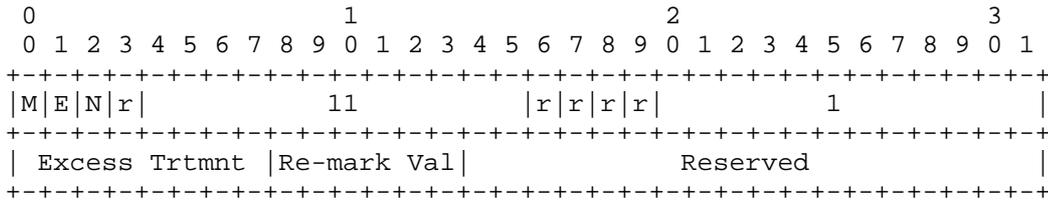
Excess Treatment: Indicates how the QNE SHOULD process out-of-profile
   traffic, that is, traffic not covered by the <TMOD> parameter.
   The Excess Treatment Parameter is set by the QNI. Allowed values
   are as follows:

   0: drop
   1: shape
   2: re-mark
   3: no metering or policing is permitted

   If no Excess Treatment Parameter is specified, the default 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, and excess traffic is
   shaped to this TMOD. The bucket size in the TMOD parameter for
   excess traffic specifies the queuing behavior, and when the
   shaping causes unbounded queue growth at the shaper, any traffic
   in excess of the TMOD for excess traffic SHOULD 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. If

excess treatment is set to 'shape' and two TMOD parameters are
specified, then the QOSM specification dictates how excess traffic
should be shaped in that case.

When excess treatment is set to 're-mark', the Excess Treatment
Parameter MUST carry the re-mark value, and the re-mark values and
procedures MUST be specified in the QOSM specification document.
For example, packets may be re-marked to pertain to a particular
QoS class (Diffserv Code Point (DSCP) value). In the latter case,
re-marking 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 re-marked to a different
QoS Class (DSCP value) indicated in the 'Re-mark Value', as
follows:

Re-mark Value (6 bits): indicates DSCP value [RFC2474] to re-mark
   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

   The coding for the <PHB Class> parameter is as follows [RFC3140]:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|E|N|r|           12          |r|r|r|r|           1           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             PHB Field          |            (Reserved)        |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```
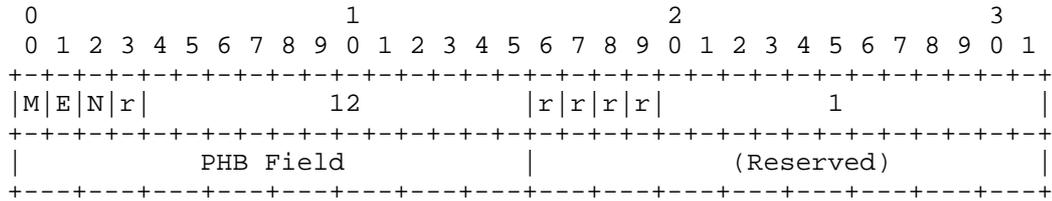
   The above encoding is consistent with [RFC3140], and the following
   four figures show four possible formats based on the value of the PHB
   Field.

   Single PHB:

```
     0                   1
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | DSCP      |0 0 0 0 0 0 0 0 0 0|
    +---+---+---+---+---+---+---+---+
```

   Set of PHBs:

```
     0                   1
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | DSCP      |0 0 0 0 0 0 0 0 1 0|
    +---+---+---+---+---+---+---+---+
```
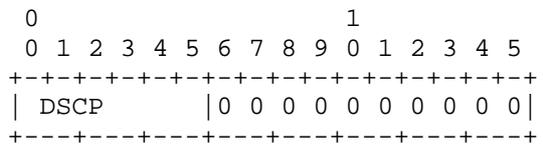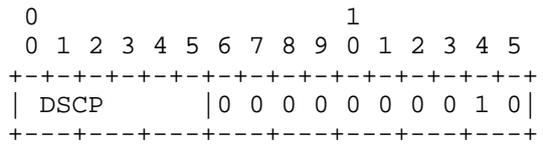
   PHBs not defined by standards action, i.e., experimental or local use
   PHBs as allowed by [RFC2474]. In this case, an arbitrary 12-bit PHB
   identification code, assigned by the IANA, is placed left-justified
   in the 16-bit field. Bit 15 is set to 1, and bit 14 is zero for a
   single PHB or 1 for a set of PHBs. Bits 12 and 13 are zero.

   Single non-standard PHB (experimental or local):

```
     0                   1
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      PHB ID CODE     |0 0 0 1|
    +---+---+---+---+---+---+---+---+
```
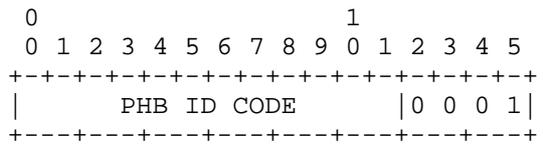
Set of non-standard PHBs (experimental or local):

```
   0                   1
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |     PHB ID CODE    |0 0 1 1|
  +---+---+---+---+---+---+---+---+
```
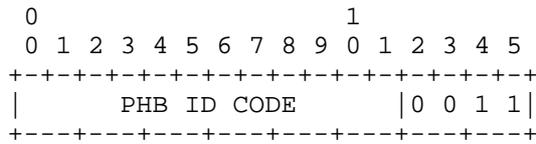
Bits 12 and 13 are reserved either for expansion of the PHB
identification code, or for other use, at some point in the future.

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] and [PHBID-CODES-REGISTRY]. Hence, no new registry
needs to be created for this purpose.

5.2.13. <DSTE Class Type> Parameter

A description of the semantic of the parameter values can be found in
[RFC4124]. The coding for the <DSTE Class Type> parameter is as
follows:

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |M|E|N|r|         13          |r|r|r|r|         1               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |DSTE Cls. Type |                (Reserved)                     |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```
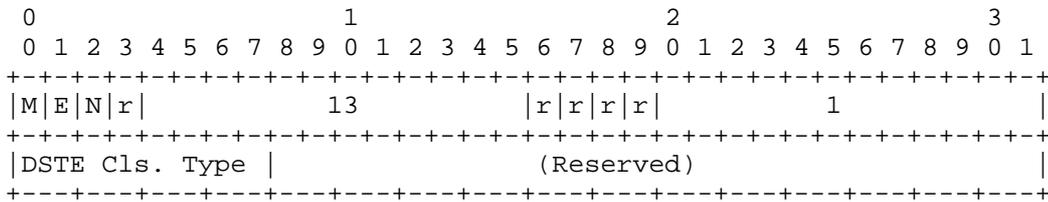
DSTE Class Type: Indicates the DSTE class type. Values currently
allowed are 0, 1, 2, 3, 4, 5, 6, and 7.

5.2.14. <Y.1541 QoS Class> Parameter

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|E|N|r|          14           |r|r|r|r|          1            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Y.1541 QoS Cls.|              (Reserved)                       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```
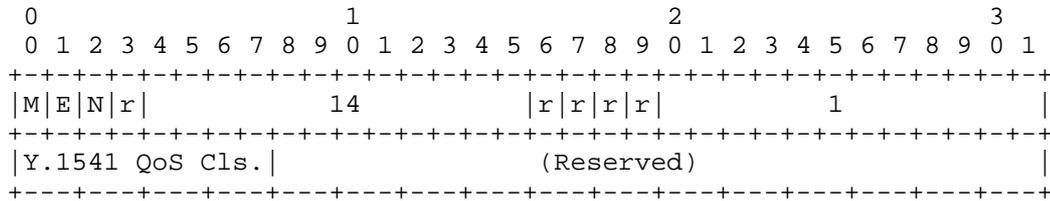
   Y.1541 QoS Class: Indicates the Y.1541 QoS Class. Values currently
   allowed are 0, 1, 2, 3, 4, 5, 6, and 7.

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

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

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

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

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

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

      Class 6:
      Applications that are highly sensitive to loss. Mean delay <= 100
      ms, delay variation <= 50 ms, and loss ratio <= 10^-5.
      Application examples include television transport, high-capacity
      TCP transfers, and Time-Division Multiplexing (TDM) circuit
      emulation.

      Class 7:
      Applications that are highly sensitive to loss. Mean delay <= 400
      ms, delay variation <= 50 ms, and loss ratio <= 10^-5.
      Application examples include 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 [RFC5974] has a very detailed security discussion in
   Section 7. All the considerations detailed in Section 7 of [RFC5974]
   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 5226
   [RFC5226]. 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 as specified in Section 5:
      0: QoS Desired
      1: QoS Available
      2: QoS Reserved
      3: Minimum QoS

   Further values are as follows:
      4-63: Unassigned
      64-67: Private/Experimental Use
      68-4095: Reserved
      (Note: 'Reserved' just means 'do not give these out'.)
   The registration procedure is Specification Required.

   QSPEC Version (4 bits):
   The following value is allocated by this specification:
      0: Version 0 QSPEC
   Further values are as follows:
      1-15: Unassigned
   The registration procedure is Specification Required. (A
   specification is required to depreciate, delete, or modify QSPEC
   versions.)

   QSPEC Type (5 bits):
   The following values are allocated by this specification:
      0: Default
      1: Y.1541-QOSM [RFC5976]
      2: RMD-QOSM [RFC5977]
   Further values are as follows:
      3-12: Unassigned
      13-16: Local/Experimental Use
      17-31: Reserved
   The registration procedure is Specification Required.

   QSPEC Procedure (8 bits):
   The QSPEC Procedure object consists of the Message Sequence parameter
   (4 bits) and the Object Combination parameter (4 bits), as discussed
   in Section 4.3. Message Sequences 0 (Two-Way Transactions), 1
   (Three-Way Transactions), and 2 (Resource Queries) are explained in
   Sections 4.3.1, 4.3.2, and 4.3.3, respectively. Tables 1, 2, and 3
   in Section 4.3 assign the Object Combination Number to Message
   Sequences 0, 1, and 2, respectively. The values assigned by this
   specification for the Message Sequence parameter and the Object
   Combination parameter are summarized here:

MSG. SEQ.	OBJ. COM.	OBJECTS INCLUDED IN QUERY MESSAGE	OBJECTS INCLUDED IN RESERVE MESSAGE	OBJECTS INCLUDED IN RESPONSE MESSAGE
0	0	N/A	QoS Desired	QoS Reserved
0	1	N/A N/A	QoS Desired QoS Available	QoS Reserved QoS Available
0	2	N/A N/A N/A	QoS Desired QoS Available Minimum QoS	QoS Reserved QoS Available
1	0	QoS Desired	QoS Desired	QoS Reserved
1	1	QoS Desired (Minimum QoS)	QoS Desired QoS Available (Minimum QoS)	QoS Reserved QoS Available
1	2	QoS Desired QoS Available	QoS Desired QoS Available	QoS Reserved
2	0	QoS Available	N/A	QoS Available

Further values of the Message Sequence parameter (4 bits) are as
follows:
    3-15: Unassigned

Further values of the Object Combination parameter (4 bits) are as
follows:

Message Sequence	Object Combination
0	3-15: Unassigned
1	3-15: Unassigned
2	1-15: Unassigned
3-15	0-15: Unassigned

The registration procedure is Specification Required. (A
specification is required to depreciate, delete, or modify QSPEC
Procedures.)

QoS Model Error Code (8 bits):
QoS Model Error Codes may be defined for NSLP error class 6 (QoS
Model Error), as described in Section 6.4 of [RFC5974]. Values are
as follows:
    0-63: Unassigned
    64-67: Private/Experimental Use

       68-255: Reserved
    The registration procedure is Specification Required. (A
    specification is required to depreciate, delete, or modify QoS Model
    Error Codes.)

    Parameter ID (12 bits):
    The following values are allocated by this specification:
    1-14: assigned as specified in Section 5.2:
       1: <TMOD-1>
       2: <TMOD-2>
       3: <Path Latency>
       4: <Path Jitter>
       5: <Path PLR>
       6: <Path PER>
       7: <Slack Term>
       8: <Preemption Priority> and <Defending Priority>
       9: <Admission Priority>
       10: <RPH Priority>
       11: <Excess Treatment>
       12: <PHB Class>
       13: <DSTE Class Type>
       14: <Y.1541 QoS Class>
    Further values are as follows:
       15-255: Unassigned
       256-259: Private/Experimental Use
       260-4095: Reserved
    The registration procedure is Specification Required. (A
    specification is required to depreciate, delete, or modify Parameter
    IDs.)

    Y.2171 Admission Priority Parameter (8 bits):
    The following values are allocated by this specification:
    0-2: assigned as specified in Section 5.2.9:
       0: best-effort priority flow
       1: normal priority flow
       2: high priority flow
    Further values are as follows:
       3-63: Unassigned
       64-255: Reserved
    The registration procedure is Specification Required.

    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 [EMERG-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. There
are no additional IANA requirements made by this specification for
the RPH Namespace Parameter.

Excess Treatment Parameter (8 bits):
The following values are allocated by this specification:
0-3: assigned as specified in Section 5.2.11:
   0: drop
   1: shape
   2: re-mark
   3: no metering or policing is permitted
Further values are as follows:
   4-63: Unassigned
   64-255: Reserved
The registration procedure is Specification Required.

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:
   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
Further values are as follows:
   8-63: Unassigned
   64-255: Reserved
The registration procedure is Specification Required.

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, 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, Karlsruhe Institute of Technology (KIT)
    Zirkel 2, Building 20.20
    P.O. Box 6980
    Karlsruhe 76049
    Germany
    Phone: +49 721 608 6413
    EMail: bless@kit.edu
    URI: http://tm.kit.edu/~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

    Bernd Schloer
    University of Goettingen
    EMail: bschloer@cs.uni-goettingen.de

    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).

   [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., Partridge, C., and R. Guerin,
                      "Specification of Guaranteed Quality of Service", RFC
                      2212, September 1997.

   [RFC2215]   Shenker, S. and J. Wroclawski, "General
                      Characterization Parameters for Integrated Service
                      Network Elements", RFC 2215, September 1997.

   [RFC3140]   Black, D., Brim, S., Carpenter, B., and F. Le
                      Faucheur, "Per Hop Behavior Identification Codes",
                      RFC 3140, June 2001.

   [RFC3181]   Herzog, S., "Signaled Preemption Priority Policy
                      Element", RFC 3181, October 2001.

   [RFC4124]   Le Faucheur, F., Ed., "Protocol Extensions for
                      Support of Diffserv-aware MPLS Traffic Engineering",
                      RFC 4124, June 2005.

   [RFC4412]   Schulzrinne, H. and J. Polk, "Communications Resource
                      Priority for the Session Initiation Protocol (SIP)",
                      RFC 4412, February 2006.

   [RFC4506]   Eisler, M., Ed., "XDR: External Data Representation
                      Standard", STD 67, RFC 4506, May 2006.

   [RFC5971]   Schulzrinne, H. and R. Hancock, "GIST: General
                      Internet Signalling Transport", RFC 5971, October
                      2010.

   [RFC5974]   Manner, J., Karagiannis, G., and A. McDonald, "NSIS
                      Signaling Layer Protocol (NSLP) for Quality-of-
                      Service Signaling", RFC 5974, October 2010.

   [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

   [COMPOSITION]   Morton, A. and E. Stephan, "Spacial Composition of
                   Metrics", Work in Progress, July 2010.

   [DQOS]   CableLabs, "PacketCable Dynamic Quality of Service
                   Specification", CableLabs Specification
                   PKT-SP-DQOS-I12-050812, August 2005.

   [EMERG-RSVP]   Le Faucheur, F., Polk, J., and K. Carlberg, "Resource
                   ReSerVation Protocol (RSVP) Extensions for Admission
                   Priority", Work in Progress, March 2010.

   [G.711]   ITU-T Recommendation G.711, "Pulse code modulation
                   (PCM) of voice frequencies", November 1988.

   [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,
                   April 2010.

   [DSCP-REGISTRY] IANA, "Differentiated Services Field Codepoints",
                   http://www.iana.org.

   [NETWORK-OCTET-ORDER]
                   Wikipedia, "Endianness",
                   http://en.wikipedia.org/wiki/Endianness.

   [PHBID-CODES-REGISTRY]
                   IANA, "Per Hop Behavior Identification Codes",
                   http://www.iana.org.

   [RFC1701]   Hanks, S., Li, T., Farinacci, D., and P. Traina,
                   "Generic Routing Encapsulation (GRE)", RFC 1701,
                   October 1994.

   [RFC1702]   Hanks, S., Li, T., Farinacci, D., and P. Traina,
                   "Generic Routing Encapsulation over IPv4 networks",
                   RFC 1702, October 1994.

   [RFC2003]   Perkins, C., "IP Encapsulation within IP", RFC 2003,
                   October 1996.

   [RFC2004]   Perkins, C., "Minimal Encapsulation within IP", RFC
                   2004, October 1996.

   [RFC2205]   Braden, R., Ed., Zhang, L., Berson, S., Herzog, S.,
                      and S. Jamin, "Resource ReSerVation Protocol (RSVP)
                      -- Version 1 Functional Specification", RFC 2205,
                      September 1997.

   [RFC2473]   Conta, A. and S. Deering, "Generic Packet Tunneling
                      in IPv6 Specification", RFC 2473, December 1998.

   [RFC2474]   Nichols, K., Blake, S., Baker, F., and D. Black,
                      "Definition of the Differentiated Services Field (DS
                      Field) in the IPv4 and IPv6 Headers", RFC 2474,
                      December 1998.

   [RFC2475]   Blake, S., Black, D., Carlson, M., Davies, E., Wang,
                      Z., and W. Weiss, "An Architecture for Differentiated
                      Service", RFC 2475, December 1998.

   [RFC2597]   Heinanen, J., Baker, F., Weiss, W., and J.
                      Wroclawski, "Assured Forwarding PHB Group", RFC 2597,
                      June 1999.

   [RFC2697]   Heinanen, J. and R. Guerin, "A Single Rate Three
                      Color Marker", RFC 2697, September 1999.

   [RFC2997]   Bernet, Y., Smith, A., and B. Davie, "Specification
                      of the Null Service Type", RFC 2997, November 2000.

   [RFC3290]   Bernet, Y., Blake, S., Grossman, D., and A. Smith,
                      "An Informal Management Model for Diffserv Routers",
                      RFC 3290, May 2002.

   [RFC3393]   Demichelis, C. and P. Chimento, "IP Packet Delay
                      Variation Metric for IP Performance Metrics (IPPM)",
                      RFC 3393, November 2002.

   [RFC3550]   Schulzrinne, H., Casner, S., Frederick, R., and V.
                      Jacobson, "RTP: A Transport Protocol for Real-Time
                      Applications", STD 64, RFC 3550, July 2003.

   [RFC3564]   Le Faucheur, F. and W. Lai, "Requirements for Support
                      of Differentiated Services-aware MPLS Traffic
                      Engineering", RFC 3564, July 2003.

   [RFC4213]   Nordmark, E. and R. Gilligan, "Basic Transition
                      Mechanisms for IPv6 Hosts and Routers", RFC 4213,
                      October 2005.

   [RFC4301]   Kent, S. and K. Seo, "Security Architecture for the

                   Internet Protocol", RFC 4301, December 2005.

   [RFC4303]   Kent, S., "IP Encapsulating Security Payload (ESP)",
                   RFC 4303, December 2005.

   [RFC5226]   Narten, T. and H. Alvestrand, "Guidelines for Writing
                   an IANA Considerations Section in RFCs", BCP 26, RFC
                   5226, May 2008.

   [RFC5481]   Morton, A. and B. Claise, "Packet Delay Variation
                   Applicability Statement", RFC 5481, March 2009.

   [RFC5976]   Ash, G., Morton, A., Dolly, M., Tarapore, P., Dvorak,
                   C., and Y. El Mghazli, "Y.1541-QOSM: Model for
                   Networks Using Y.1541 Quality-of-Service Classes",
                   RFC 5976, October 2010.

   [RFC5977]   Bader, A., Westberg, L., Karagiannis, G., Kappler, C,
                   and T. Phelan, "RMD-QOSM: The NSIS Quality-of-Service
                   Model for Resource Management in Diffserv", RFC 5977,
                   October 2010.

   [VERTICAL-INTERFACE]
                   Dolly, M., Tarapore, P., and S. Sayers, "Discussion
                   on Associating of Control Signaling Messages with
                   Media Priority Levels", T1S1.7 and PRQC, October
                   2004.

   [Y.1540]   ITU-T Recommendation Y.1540, "Internet Protocol Data
                   Communication Service - IP Packet Transfer and
                   Availability Performance Parameters", December 2002.

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 that includes a Receiver TSpec describing
   the resources reserved using the same parameters as the Sender TSpec,
   as well as an RSpec that provides additional IntServ QoS Model
   specific parameters, e.g., Rate and Slack.

   The RSVP TSpec, AdSpec, and RSpec are tailored to the receiver-
   initiated signaling employed by RSVP and the IntServ QoS Model. For
   example, 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.   Example of TMOD Parameter Encoding

   In an example VoIP application that uses RTP [RFC3550] and the G.711
   Codec [G.711], the TMOD-1 parameter could be set as follows:

   In the simplest case, the Minimum Policed Unit m is the sum of the
   IP, UDP, and RTP headers + payload. The IP header in the IPv4 case
   has a size of 20 octets (40 octets if IPv6 is used). The UDP header
   has a size of 8 octets, and RTP uses a 12-octet header. The G.711
   Codec specifies a bandwidth of 64 kbit/s (8000 octets/s). Assuming
   RTP transmits voice datagrams every 20 ms, the payload for one
   datagram is 8000 octets/s * 0.02 s = 160 octets.

   IPv4 + UDP + RTP + payload: m = 20 + 8 + 12 + 160 octets = 200 octets
   IPv6 + UDP + RTP + payload: m = 40 + 8 + 12 + 160 octets = 220 octets

   The Rate r specifies the amount of octets per second. 50 datagrams
   are sent per second.

   IPv4: r = 50 1/s * m = 10,000 octets/s
   IPv6: r = 50 1/s * m = 11,000 octets/s

The bucket size b specifies the maximum burst. In this example, a
burst of 10 packets is used.

IPv4: b = 10 * m = 2000 octets
IPv6: b = 10 * m = 2200 octets

A number of extra headers (e.g., for encapsulation) may be included
in the datagram. A non-exhaustive list is given below. For
additional headers, m, r, and b have to be set accordingly.

```
Protocol Header Size
-------------------------+------------
GRE [RFC1701]            |    8 octets
GREIP4 [RFC1702]         |  4-8 octets
IP4INIP4 [RFC2003]       |   20 octets
MINENC [RFC2004]         | 8-12 octets
IP6GEN [RFC2473]         |   40 octets
IP6INIP4 [RFC4213]       |   20 octets
IPsec [RFC4301, RFC4303] |    variable
-------------------------+------------
```

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)
   ck technology concepts
   Berlin, Germany
   EMail: cornelia.kappler@cktecc.de

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