Network WG
Internet-Draft
Intended status: Proposed Standard
Expires: August 25, 2013
Updates: RFC 2205 & 4495 (if published as an RFC)

James Polk
Subha Dhesikan
Cisco
February 25, 2013

Resource Reservation Protocol Multiple Instance Object
draft-polk-rsvp-multi-instance-object-01.txt

Abstract

This document creates the framework for a new Resource Reservation Protocol version 1 (RSVP) object for instances in which there are multiple occurrences of existing RSVP objects is to be included within the same RSVP message. This document offers two instances for multiple versions of the same object will be valid in RSVP messages, for more than one traffic specification object (TSPEC), and more than one TSPEC priority object (PREEMPTION_PRI).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 25, 2013.

Copyright Notice

Table of Contents

1. Introduction

This document creates the framework for a new Resource Reservation Protocol version 1 (RSVP) [RFC2205] object for instances in which there is a need to carry multiple occurrences of included RSVP object within the same RSVP message. The need for multiple versions of existing objects is for environments in which the information conveyed within these objects may or may not be grantable by the network. To optimization this operation, if a different version of the same object, with different information or demands, can be included without the need for that rejecting entire RSVP message. For example, the initial RSVP PATH message contains a request for a 12Mbps bandwidth reservation, but that amount is not grantable by one or more network nodes. If a reduced amount of bandwidth can still granted, and is acceptable to the network as well as both endpoints, allowing that PATH message to contain a backup bandwidth request for, say 4Mbps, saves the time of completely rejecting the initial PATH and having the sender generate a new PATH. A complete rejection to this scenario is how RSVP operates today.

This is a general purpose optimization for RSVP, and will allow any RSVP object to have multiple versions of an existing object, provided that existing object is specified to do so. It is important to understand that RSVP operates normally, with all Objects and elements in their native locations. This document offers two instances for multiple versions of the same object will be valid in RSVP messages, for more than one traffic specification object (TSPEC) [RFC2210], and more than one priority element (PREEMPTION_PRI) [RFC3181]. This extension will bring RSVP more in line with existing application layer protocols that offer multiple choices for the specifics within a call or session. At the same

time, all extra versions of Objects or elements are contained in a single location that is ignored if not understood. Thus, backwards compatibility is assured.

Realtime session set-up protocols such as SIP [RFC3261] carry a Session Description Protocol (SDP) [RFC4566] payload which establishes the parameters for rich media calls (i.e., voice, video) between two or more endpoints. Since the late 1990s, SDP has had the capability to offer more than one codec per application type (i.e., more than one audio payload type and/or more than one video payload type), which can be carried in the same session set-up message. This means a calling endpoint can give the called party a list of codecs to choose from for that call, as well as multiple applications for that call.

With this RSVP extension, for example, a SIP voice and/or video call can have a reservation adapt to whichever codec(s) are picked for that call, without wasting unnecessary bandwidth that will not be utilized.

Visually, Figure 1. is a normal RSVP reservation set-up exchange that is accepted by all RSVP enabled nodes.

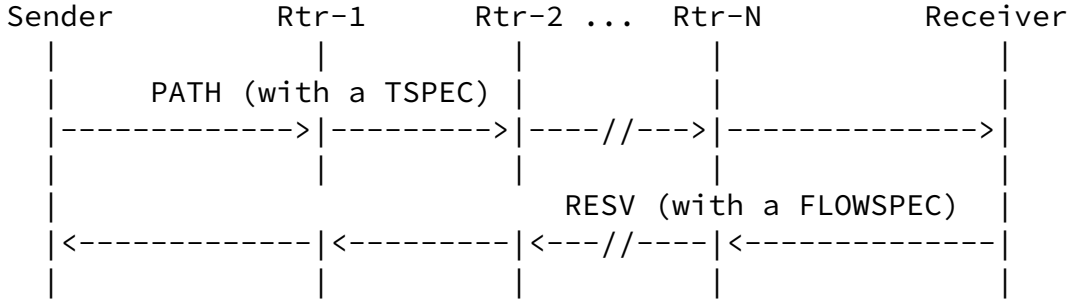


Figure 1. Concept of RSVP in a Single Direction

However, Figure 2. is a normal RSVP reservation set-up exchange that is rejected as the reservation is partially established. We will use bandwidth as the reason for the rejection because it is probably the easiest thing to understand about RSVP, that it creates reservation of fixed bandwidth.

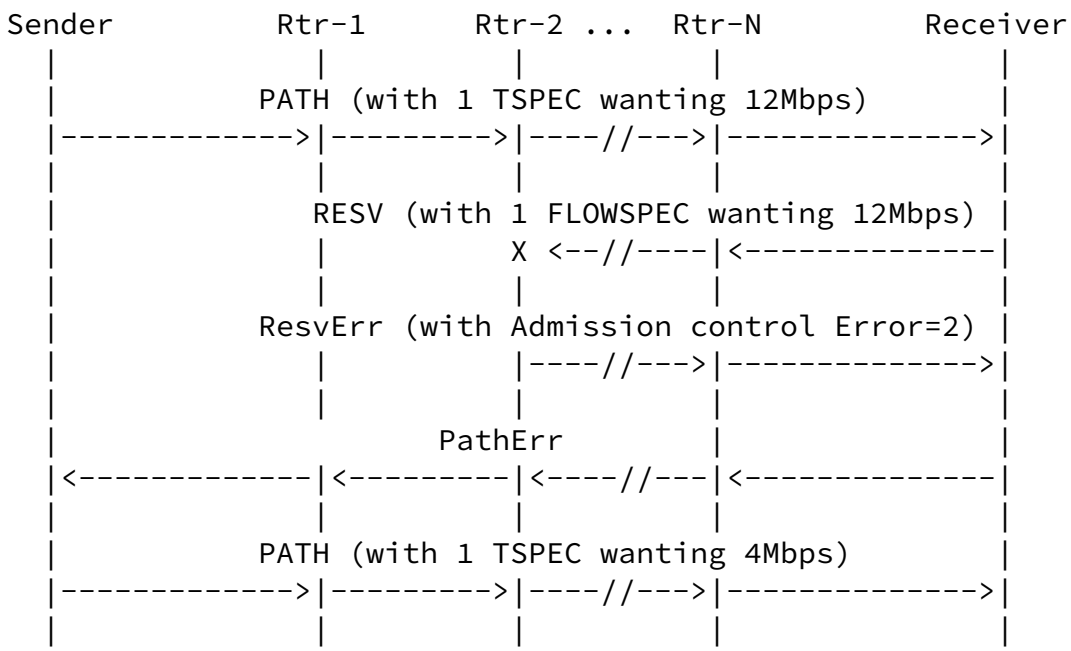


Figure 2. Concept of RSVP Rejection due to Limited Bandwidth

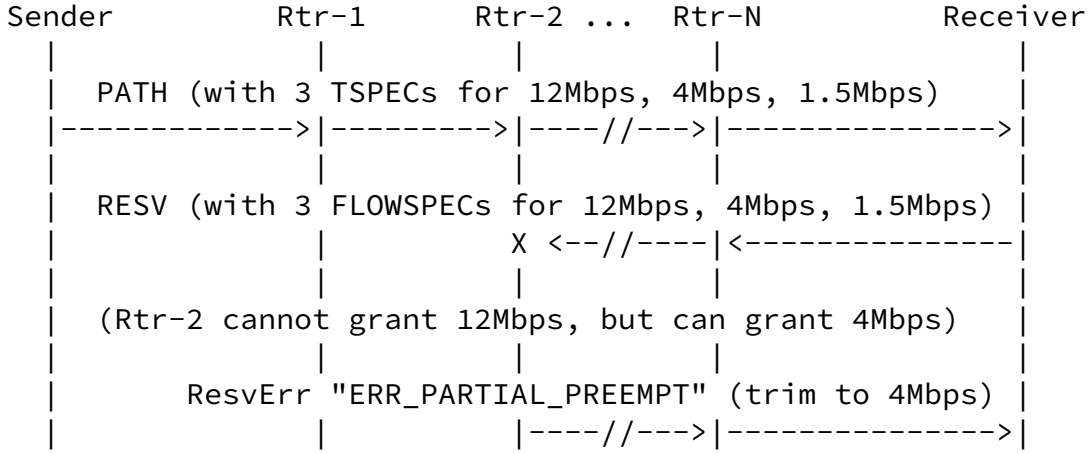
Rtr-2 in the above example reservation attempt rejects the bandwidth requested for this reservation. Once the Sender receives

the PathErr message indicating why the rejection occurred, it can attempt a new reservation requesting less bandwidth. Regrettably, this is a bit of a guessing game put on the Sender to figure out how much bandwidth to request next. When this scenario is complicated when the reservation request is initiated because of a layer 7 signaling protocol, such as SIP, to establish a call between two endpoints, as defined in [RFC3312]. All the users experience is further delay as RSVP attempts to successfully establish the reservation before "the phone can ring".

Presently, translating a (SIP) layer 7 operation into RSVP at layer 4, only a single reservation can be established per application (i.e., one for voice, one for video) at a time (without creating chaos). This one reservation request would most probably its bandwidth request using the codec with the largest bandwidth requirements. Bandwidth parameters are conveyed within a traffic specification (TSPEC), as defined in [RFC2210]. Once one considers the bandwidth needs of present day video codecs, always initially setting up the maximum bandwidth reservation is less than optimal (some might argue criminal).

If, on the other hand, the initial RSVP PATH message could contain more than one version of a TSPEC, say one per codec. Then the reservation would be established with the greatest amount of bandwidth the network could grant at its most congested node in the signaling path, which would in turn choose for the endpoints which codec within SDP is selected for this call.

Thus, this extension would solve the problem in Figure 2. in this way,



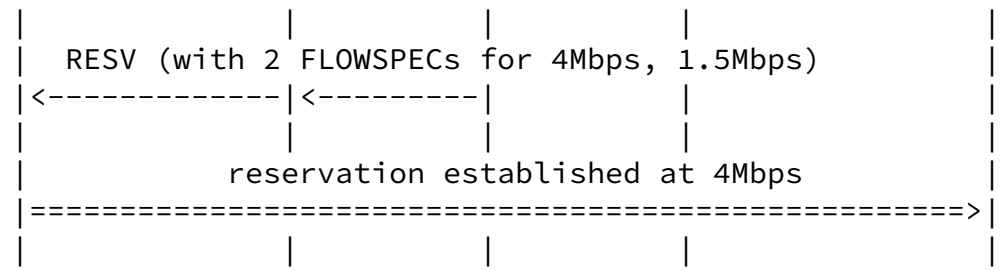


Figure 3. Concept of RSVP Rejection due to Limited Bandwidth

Figure 3. shows a RSVP PATH message containing 3 TSPECs (12Mbps, 4Mbps, and 1.5Mbps), placed in that order in the PATH. Router 2 (Rtr-2) cannot grant the RESV message at 12Mbps, but can grant the 4Mbps bandwidth request. Rtr-2 trims the bandwidth upstream with a slight modification to the procedures defined in RFC 4495, and transmits the RESV downstream without the 12Mbps bandwidth request, which was removed from the RESV. The Sender, in this example, receives the RESV with 4Mbps and the reservation is established.

In Section 3., we will create the framework and format for the MULTI_INSTANCE Object. In Section 4., we will show how to include multiple TSPEC Objects within this MULTI_INSTANCE Object, as well as stipulate the rules for TSPEC usage. In Section 5., we will show how to include multiple PREEMPTION_PRI Objects within this MULTI_INSTANCE Object, as well as stipulate the rules for PREEMPTION_PRI usage. Section 6. will have the IANA registry considerations, and Section 7. will have the Security considerations.

2. Terminology

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 [RFC2119] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

3. Framework for the MULTI_INSTANCE Object

The format of all RSVP Objects is based on a series of 32-bit words. This is true with the MULTI_INSTANCE Object as well. Normally, RSVP

and IntServ documents specify Objects in a 32-bit wide, top down format, where the most significant bit is the top left bit, and the least significant bit on the right. This is loosely shown in Figure 4. below.

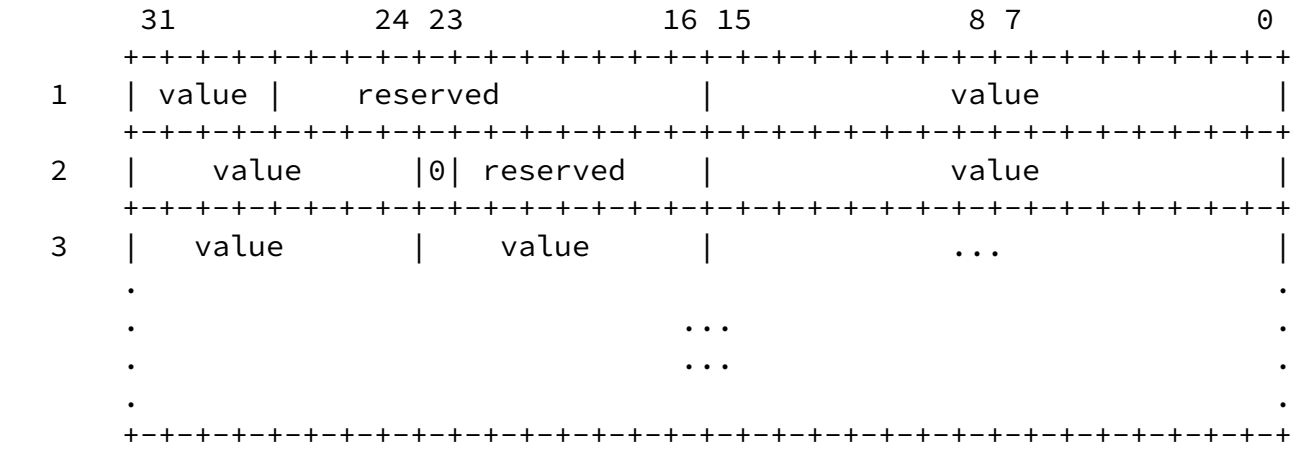


Figure 4. Generic RSVP Format for Illustration Purposes

The individual field value lengths within each RSVP Object depend on the Object, thus Figure 4. is merely an example (which happens to be the first 3 words format of a TSPEC).

However, RSVP messages can be quite long in this format, so what one usually sees in documents are each individual Object and no overall RSVP message format. Each Object has an field identifier indicating which RSVP message this Object is within (e.g., PATH, RESV, REFRESH), as well as a 'Parameter ID' indicating which Object within this (e.g., TSPEC, Rspec, Policy_Data).

Looking at RSVP another way, to illustrate the point about where certain parts can be within an overall RSVP message, Figure 5. Shows an example RSVP message on its side, where the top of the message is on the left and the bottom of the message is on the right. With that in mind, the most significant bit of the top 32-bit word is on the lower left of Figure 5., and the least significant bit is on the upper left. The length of this message can vary and does not represent anything other than this message has some size to it; i.e., it has a number of Objects within this message, including a sender_descriptor where the 'primary' TSPEC Object resides, and Policy_Data Object where the PREEMPTION_PRI Object resides. Additionally in the RSVP message is the proposed MULTI_INSTANCE Object, which is neither in the sender_descriptor or Policy_Data Object.

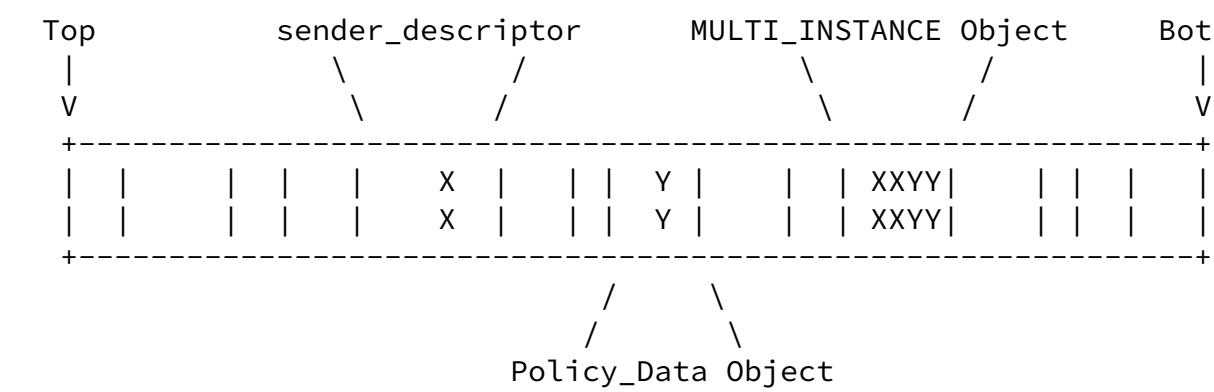


Figure 5. Generic RSVP Format Shown Sideways

It is important to remember that RSVP enabled nodes will always ignore Objects that are not understood. This allows the protocol to be extended before all the RSVP nodes are upgraded to understand new functions and capabilities. In other words, no one expects a single 'flag day' upgrade to occur in all routers at the same time in the same network, which could be disruptive if not performed correctly.

An important aspect of this new Object is that the initial copy or instance of the Object, however many Objects have multiple instances in this RSVP message, MUST remain in its original place within the message. We will refer to this original version of an Object or element to be the 'primary' version or copy. Its placement allows RSVP to operate normally. The MULTI_INSTANCE Object only carries a second, third, etc. versions of Objects. Once the RSVP node determines that it cannot grant what is asked for in an existing Object, it will look to the MULTI_INSTANCE Object for the next instance of that Object to replace the original with. Failing this, the RSVP message will mostly likely be rejected through the normal procedures already defined in RSVP documentation.

To give a practical example of this, we will use the message flow from Figure 3. In it, we have a PATH message carrying not one, but three TSPECs for 12Mbps, 4Mbps and 1.5Mbps. Once Rtr-2 cannot grant the primary TSPEC asking for 12Mbps, that router discards that TSPEC, from the RSVP message. It knows to look into the MULTI_INSTANCE Object for a second version of the TSPEC. Finding two (4Mbps and 1.5Mbps), Rtr-2 moves the 4Mbps TSPEC completely into the sender_descriptor as the new 'primary' TSPEC and attempts to establish the reservation at 4Mbps. In this case, 4Mbps is granted and transmits the RESV upstream towards Rtr-1 with only one remaining TSPEC in the MULTI_INSTANCE Object.

[EDITOR'S NOTE: It is important to state that, so far, we have not defined where this MULTI_INSTANCE Object goes

within the RSVP message.]

The MULTI_INSTANCE Object can have a number of versions of the same Object that is within the RSVP message, as well as can have more than one different type of Object. To this end, here is the proposed

generic format for the MULTI_INSTANCE Object that is only carrying a single other Object

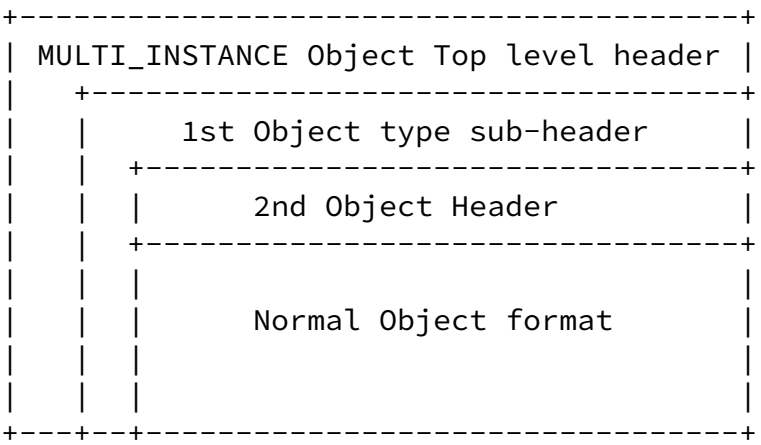
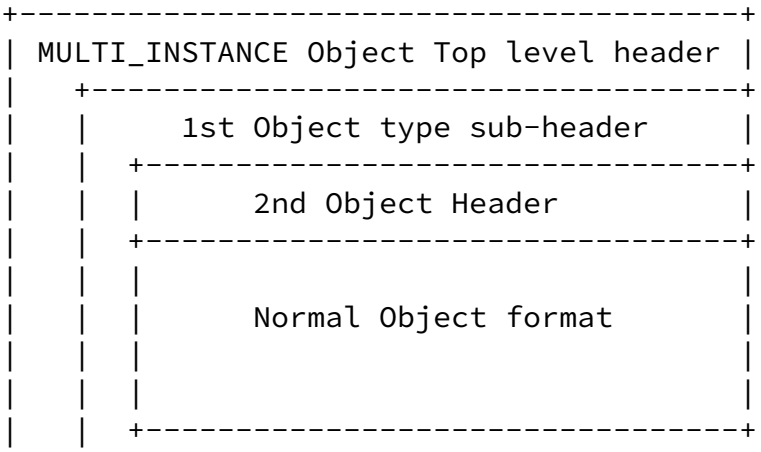


Figure 6. MULTI_INSTANCE with 1 Object

Figure 6. shows a complete second version of an existing RSVP Object, which can be removed and copied bit-for-bit into the normal placement of this Object within the RSVP message. It is important to note that this MUST be a complete new copy of a valid Object.

Figure 7. shows a MULTI_INSTANCE Object with a second and third version of the same RSVP Object



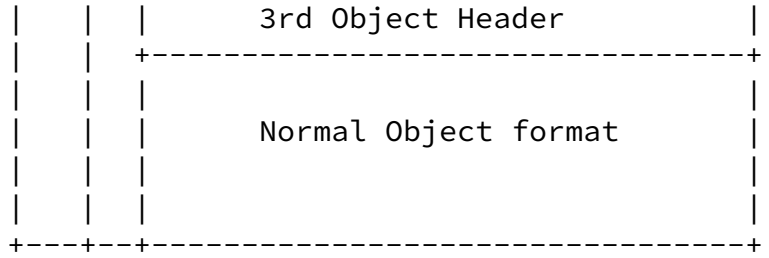


Figure 7. MULTI_INSTANCE with 2 Objects

Again, version 2 and 3 are completely valid versions of the RSVP Object they are meant to replace, with no change in any value allowed.

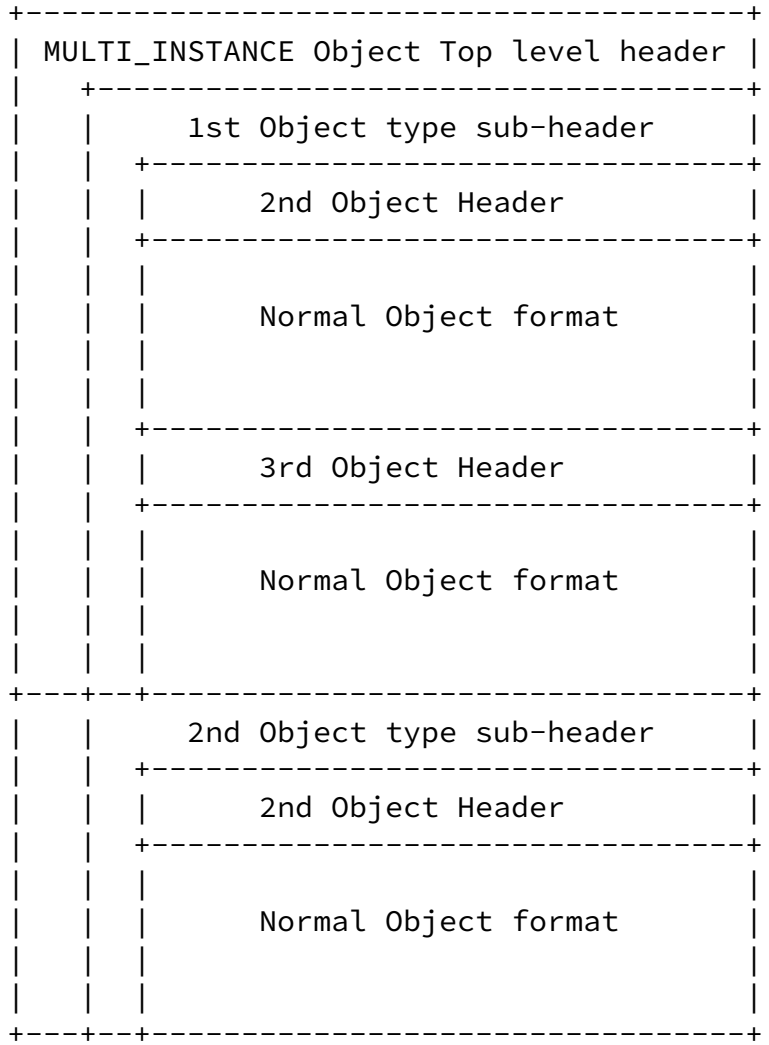


Figure 8. MULTI_INSTANCE with 2 Objects

Figure 8. adds a second type of Object to the MULTI_INSTANCE Object that is shown in Figure 7. To be able to add another type of Object, and not a second copy of the same Object, a new Object Type header is REQUIRED to preface the first 32-bit word of the new Object. These two different Objects carried within the MULTI_INSTANCE Object can be related, or they might not have anything to do with each other.

3.1 The MULTI_INSTANCE Object Format

The multi-32-bit word format of the MULTI_INSTANCE Object is TBD in a subsequent revision of this document.

3.2 Rules for building a MULTI_INSTANCE Object

The following are the rules for implementations of the MULTI_INSTANCE object:

#1 - having only 1 *SPEC or Object is allowed in the MULTI_INSTANCE

Object (i.e., a grouping can have a single entry)

#2 - more than one *SPEC or Object is allowed in the MULTI_INSTANCE Object (i.e., separate groups can have a single entry each)

#3 - more than one *SPEC or Object is allowed in the MULTI_INSTANCE Object (i.e., separate groups can have a multiple entries each)

#4 - some groupings within MULTI_INSTANCE MUST be paired in whenever a single instance occurs in any group.

In other words, based on rule #3, if a TSPEC is in each group, so MUST there be an RSPEC if any RSPEC is within this MULTI_INSTANCE Object. An RSPEC is an example of a *SPEC that MUST NOT be alone without its TSPEC.

4. Multiple TSPEC Objects in the MULTI_INSTANCE Object

This document defines the framework for the MULTI_INSTANCE Object,

as well as for two Objects to be available for inclusion within this new Object: the TSPEC Object and the PREEMPTION_PRI Object (detailed in Section 5.). This section deals with how to include one or more TSPEC Objects within the MULTI_INSTANCE Object.

This document specifies if the reservation is to be Controlled Load [RFC2211], the entire TSPEC, including the two 32-bit word headers (totaling eight 32-bit words), are included in the MULTI_INSTANCE object. An example of a TSPEC from RFC 2210 is here in Figure 9.:

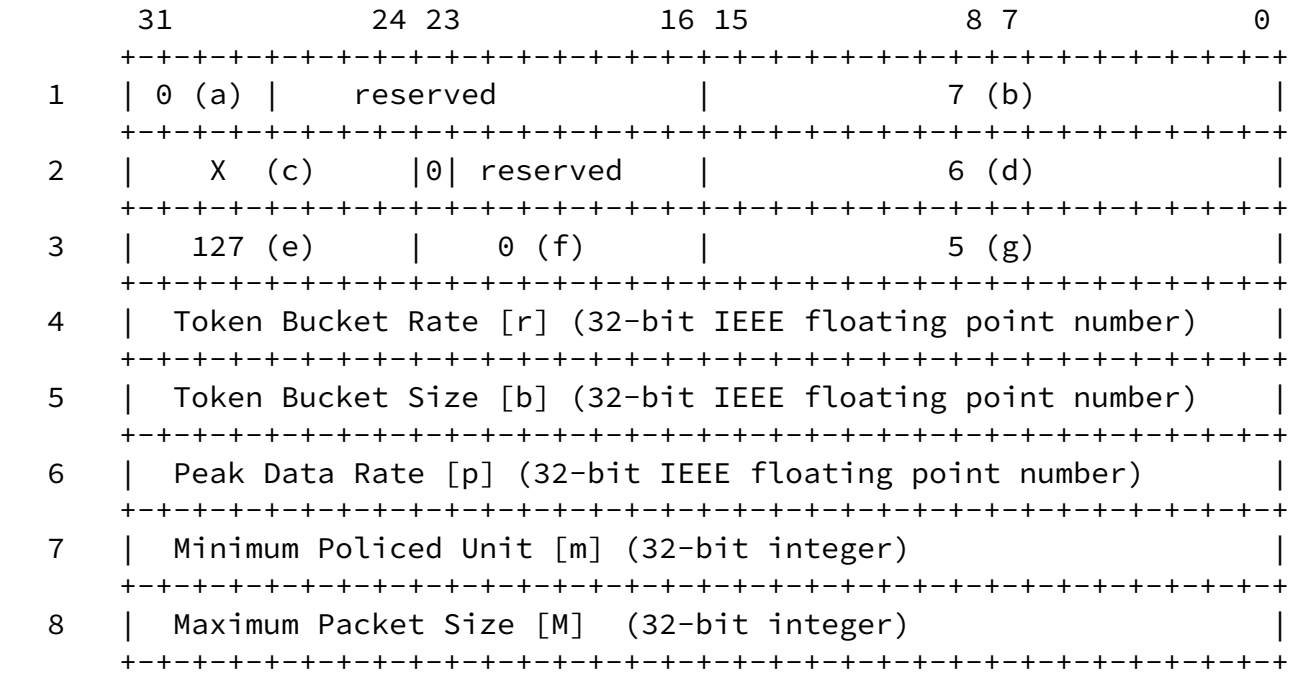


Figure 9. Controlled Load SENDER_TSPEC in a PATH

(a) - Message format version number (0)
(b) - Overall length (7 words not including header)
(c) - Service header, service number
- '1' (Generic information) if in a PATH message;
(d) - Length of service data, 6 words not including per-service header
(e) - Parameter ID, parameter 127 (Token Bucket TSpec)
(f) - Parameter 127 flags (none set)
(g) - Parameter 127 length, 5 words not including per-service header

This document specifies if the reservation is to be Guaranteed Service, the entire TSPEC and RSPEC, including the two 32-bit word headers (totaling eleven 32-bit words), are included in the MULTI_INSTANCE object as a single consecutive chunk.

A request guaranteed service reservation contains a TSPEC and RSPEC [RFC2215], as shown in Figure 10.:

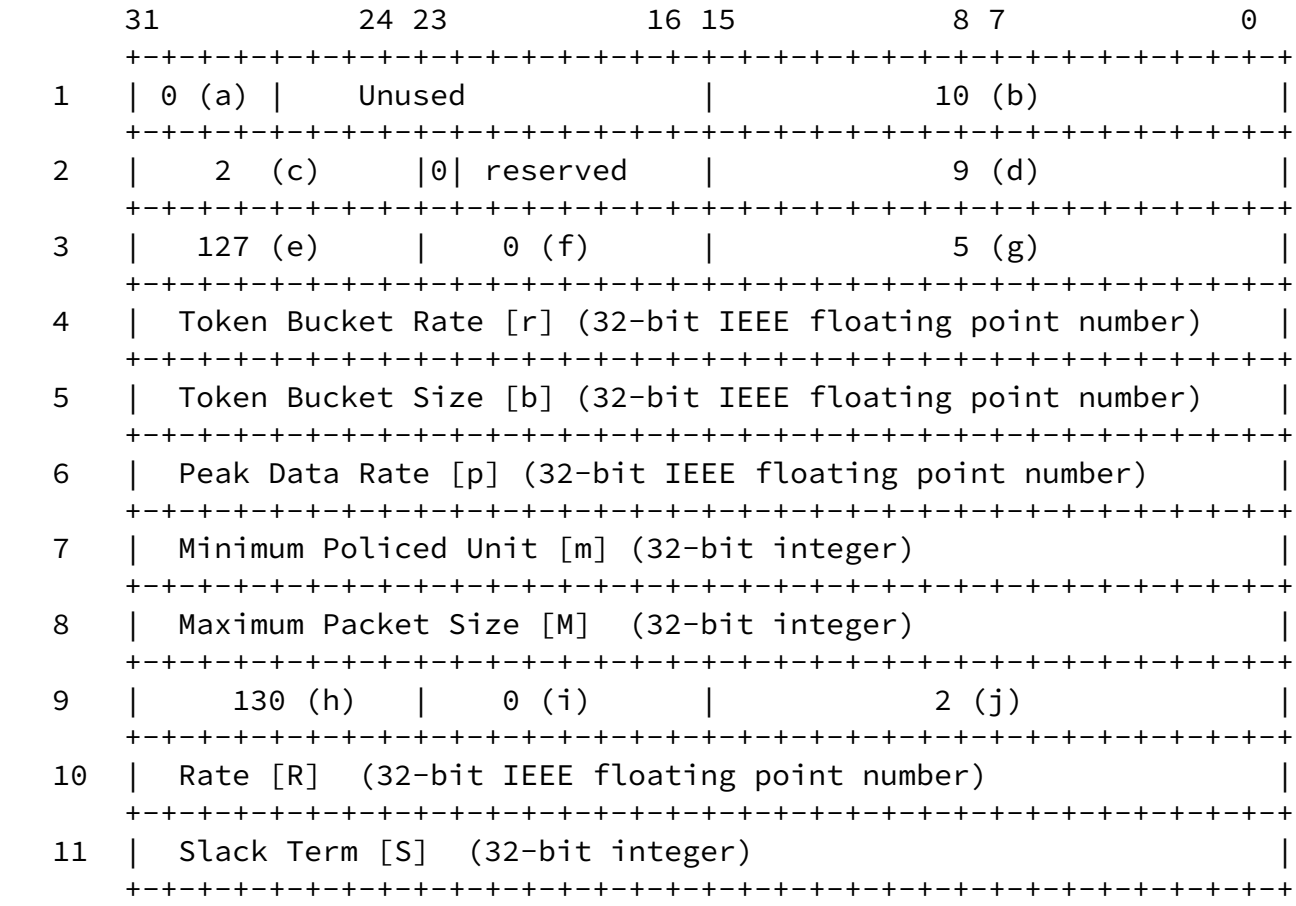


Figure 10. Guaranteed Service SENDER_TSPEC in a PATH

(a) - Message format version number (0)
(b) - Overall length (9 words not including header)
(c) - Service header, service number 2 (Guaranteed)
(d) - Length of per-service data, 9 words not including per-service header

(e) - Parameter ID, parameter 127 (Token Bucket TSpec)
(f) - Parameter 127 flags (none set)

(g) - Parameter 127 length, 5 words not including parameter header
(h) - Parameter ID, parameter 130 (Guaranteed Service RSpec)
(i) - Parameter xxx flags (none set)
(j) - Parameter xxx length, 2 words not including parameter header

The difference in structure between the Controlled-Load FLOWSPEC and Guaranteed FLOWSPEC is the RSPEC, defined in [RFC2212]. The difference with respect to the MULTI_INSTANCE Object is found in the first 32-bit word, value 'b' above - the TSPEC Object overall length. This will tell a node whether it is a Controlled Load or Guaranteed Service TSPEC.

As a reminder, TSPECs contained in the MULTI_INSTANCE Object MUST NOT be altered when moved from the MULTI_INSTANCE Object to the sender_descriptor or FLOWSPEC. Generically, this needs to be a simple cut and paste operation.

If there are multiple TSPECs in the MULTI_INSTANCE Object, each MUST be the same type of TSPEC. In other words, there MUST NOT be a mix of Controlled Load with Guaranteed Service TSPECs in the same MULTI_INSTANCE Object.

RFC 4495 defines how existing reservations can partially preemption (trim) the agreed upon bandwidth assigned to an existing reservation. This specification extends RFC 4495 by allowing that trimming of bandwidth assigned to a reservation to occur during reservation establishment downstream. This occurs when a node upstream cannot grant the bandwidth already granted downstream, but that upstream node can grant a reduced amount of bandwidth from another TSPEC within FLOWSPEC, from within the MULTI_INSTANCE Object. This operation is shown in Figure 3.

5. Multiple PREEMPTION_PRI Elements in the MULTI_INSTANCE Object

The order of the TSPECs within the MULTI_INSTANCE Object is one way to determine which is the next TSPEC to be processed by a router. Another way of determining which TSPEC is the next one to be processed is by allowing the dynamic bandwidth selection to reflect a different reservation priority for each of the multiple "bandwidth" associated with a reservation.

[RFC2750] presents a set of extensions for supporting generic policy based admission control in RSVP. These extensions include the standard format of POLICY_DATA objects, and a description of RSVP's handling of policy events. These extensions are consistent with the framework for policy-based admission control presented in [RFC2753]. POLICY_DATA objects are carried by RSVP messages and contain policy information. The exchange of POLICY_DATA objects between policy-capable nodes along the data path, supports the generation

and enforcement of consistent end-to-end admission control policies.

POLICY_DATA objects contain a list of Policy Elements that each contain a single unit of information necessary for the evaluation of policy rules. Multiple policy elements are already specified. For example, [RFC2872] specifies the Application and Sub Application Identity policy element for use with RSVP.

[RFC3181] specifies another policy element, the Preemption Priority Policy Element, that can be signaled in RSVP so that network node may take into account this policy element in order to preempt some previously admitted low priority sessions in order to make room for a newer, higher priority session. The Preemption Priority Policy Element (PREEMPTION_PRI) contains:

- one Preemption Priority specifying the priority of the new flow compared with the defending priority of previously admitted flows.

- one Defending Priority that is used once this reservation is established to compare with the preemption priority of new flows.

The format of preemption priority policy element (copied from RFC 3181) is as follows:

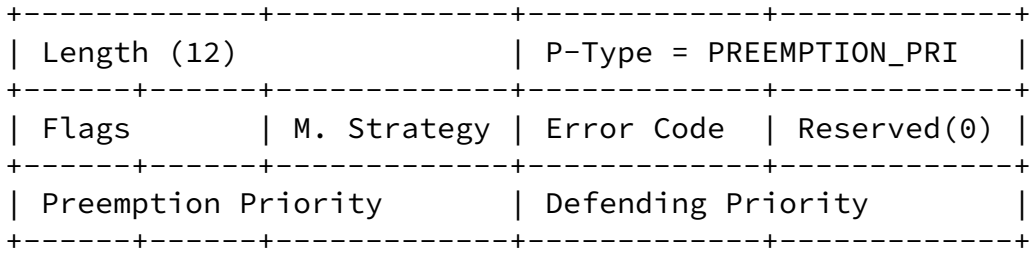


Figure 11. Preemption Priority Policy Element Format

Length: 16 bits
Always 12. The overall length of the policy element, in bytes.

P-Type: 16 bits
PREEMPTION_PRI = 1

This value is registered with IANA, see Section 7.

Flags: 8 bits

Reserved (always 0).

Merge Strategy: 8 bit
1 Take priority of highest QoS: recommended
2 Take highest priority: aggressive
3 Force Error on heterogeneous merge

Reserved: 8 bits
Error code: 8 bits
0 NO_ERROR Value used for regular PREEMPTION_PRI elements
1 PREEMPTION This previously admitted flow was preempted
2 HETEROGENEOUS This element encountered heterogeneous merge

Reserved: 8 bits
Always 0.

Preemption Priority: 16 bit (unsigned)
The priority of the new flow compared with the defending priority of previously admitted flows. Higher values represent higher Priority.

Defending Priority: 16 bits (unsigned)
Once a flow was admitted, the preemption priority becomes irrelevant. Instead, its defending priority is used to compare with the preemption priority of new flows.

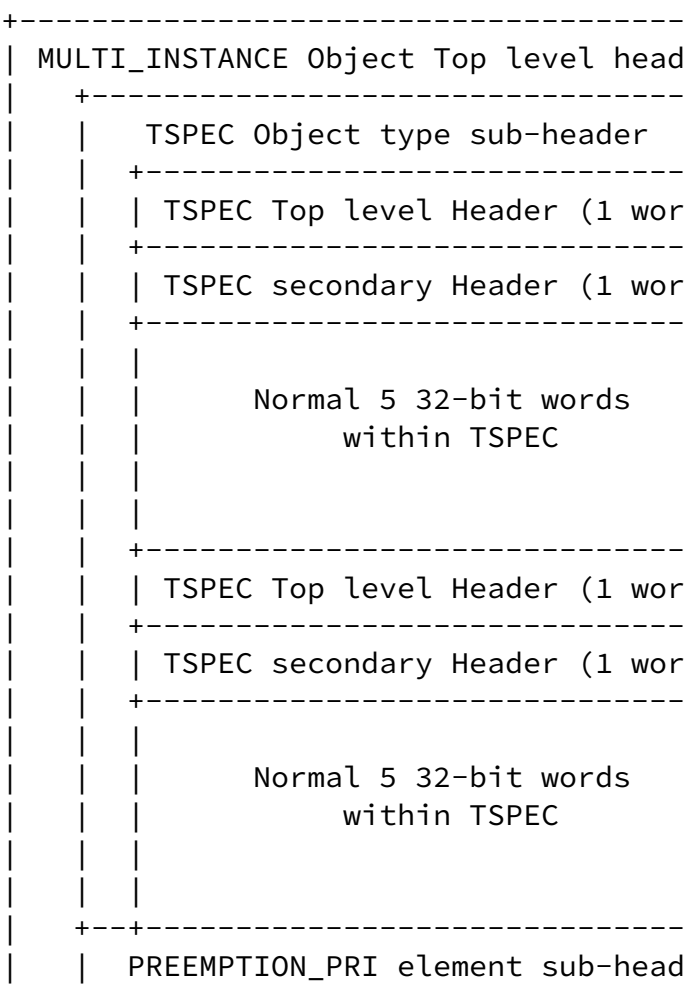
For any specific flow, its preemption priority MUST always be less than or equal to the defending priority.

The preemption priority and defending priority of the Preemption Priority Policy Element carried in a RESV message MUST be associated with the flow specification carried in the FLOWSPEC object. There MUST be either no Preemption Priority Policy Element carried in the MULTI_INSTANCE Object, or there needs to be the exact same number of Preemption Priority Policy Element as there are TSPEC Objects. This MUST be a one-to-one mapping of numbers. For example, the preemption priority and defending priority of the first (respectively second) sub-element (when present) of the MULTI_INSTANCE Object is to be associated with the first (respectively second) flow specification (when present) in the MULTI_INSTANCE Object.

If a RESV message contains a dissimilar number of TSPECs than

Preemption Priority Policy Elements in the MULTI_INSTANCE object, but contains a Preemption Priority Policy Element in the POLICY_DATA object, then the Preemption Priority Policy Element in the MULTI_INSTANCE object MUST be ignored, and all TSPECs retain the priority properties of the Preemption Priority Policy Element in the POLICY_DATA object.

An example MULTI_INSTANCE Object with 2 TSPEC Objects and 2 Preemption Priority Policy Element is showing generically in Figure 12.



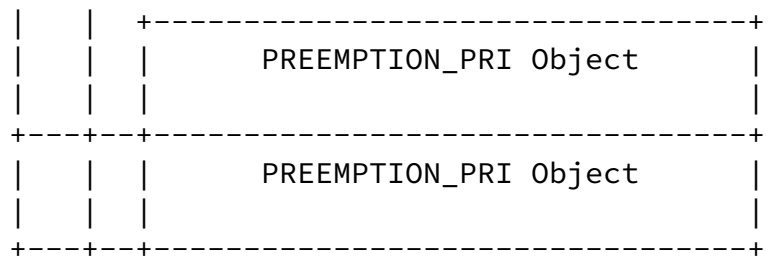


Figure 12. MULTI_INSTANCE with 2 TSPECs and 2 PREEMPTION_PRIs

6. IANA Considerations

This document IANA registers the following new parameter name in the rsvp-parameters assignments at [IANA]:

Registry Name: Parameter Names
Registry:

Value	Description	Reference
125	Multiple_Instance_object	[RFCXXXX]

Where RFCXXXX is replaced with the RFC number assigned to this Document.

This document IANA registers the following new error subcode in the

Error code section, under the Admission Control Failure (error=1), of the rsvp-parameters assignments at [IANA]:

Registry Name: Error Codes and Globally-Defined Error Value Sub-Codes
Registry:
"Admission Control Failure"

Error Subcode		meaning	Reference
6	=	MULTI_INSTANCE bandwidth unavailable	[RFCXXXX]

7. Security Considerations

The security considerations for this document do not exceed what is already in RFC 2205 (RSVP), as nothing in either of those documents

prevent a node from requesting a lot of bandwidth in a single TSPEC, or what priority values are given in a Preemption Priority Policy Element. This document merely reduces the signaling traffic load on the network by allowing many requests that fall under the same policy controls to be included in a single round-trip message exchange.

Further, this document does not increase the security risk(s) to that defined in RFC 4495, where this document creates additional meaning to the RFC 4495 created error code 102.

A misbehaving Sender can include too many TSPECs in the MULTI_INSTANCE object, which can lead to an amplification attack. That said, a bad implementation can create a reservation for each TSPEC received from within the RESV message. The number of TSPECs in the new MULTI_INSTANCE object is limited, and the spec clearly states that only a single reservation is to be set up per RESV message.

To ensure the integrity of RSVP, the RSVP Authentication mechanisms defined in [RFC2747] and [RFC3097] SHOULD be used. Those protect RSVP message integrity hop-by-hop and provide node authentication as well as replay protection, thereby protecting against corruption and spoofing of RSVP messages.

8. Contributing Authors

The authors here would like to thank the authors of draft-lefaucheur-tsvwg-rsvp-multiple-preemption-02 for allowing that draft to be merged with this draft, specifically for the Preemption Priority Policy Element discussion in Section 5. They are:

Francois Le Faucheur
Arun Kudur and
Ashok Narayanan

9. Acknowledgements

The authors wish to thank Fred Baker, Joe Touch, Bruce Davie, Dave Oran, Ashok Narayanan, Lou Berger, Lars Eggert, Arun Kudur, Janet Gunn and Ken Carlberg for their helpful comments and guidance in this effort.

10. References

10.1 Normative References

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997

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

[RFC2210] J. Wroclawski, "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997

[RFC2211] J. Wroclawski, "Specification of the Controlled-Load Network Element Service ", RFC 2211, September 1997

[RFC2212] S. Shenker, C. Partridge, R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997

[RFC2215] S. Shenker, J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements", RFC 2212, September 1997

[RFC2747] F. Baker, B. Lindell, M. Talwar, " RSVP Cryptographic Authentication", RFC2747, January 2000

[RFC2750] S. Herzog, "RSVP Extensions for Policy Control", RFC 2750, January 2000

[RFC2753] R. Yavatkar, D. Pendarakis, R. Guerin, "A Framework for Policy-based Admission Control", RFC 2753, January 2000

[RFC2872] Y. Bernet, R. Pabbati, "Application and Sub Application Identity Policy Element for Use with RSVP", RFC 2872, June 2000

[RFC3097] R. Braden, L. Zhang, "RSVP Cryptographic Authentication -- Updated Message Type Value", RFC 3097, April 2001

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

[RFC3261] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, May 2002.

[RFC3312] G. Camarillo, Ed., W. Marshall, Ed., J. Rosenberg, "Integration of Resource Management and Session Initiation Protocol (SIP)", RFC 3312 Preconditions, October 2002

[RFC4495] J. Polk, S. Dhesikan, "A Resource Reservation Protocol (RSVP) Extension for the Reduction of Bandwidth of a Reservation Flow", RFC 4495, May 2006

[RFC4566] M. Handley, V. Jacobson, C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006

10.2 Informative References

draft-lefaucheur-tsvwg-rsvp-multiple-preemption

draft-ietf-tsvwg-intserv-multiple-tspec

Author's Addresses

James Polk
3913 Treemont Circle
Colleyville, Texas, USA
+1.817.271.3552

mailto: jmpolk@cisco.com

Subha Dhesikan
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134 USA

mailto: sdhesika@cisco.com

Appendix A - History

This history of how we got to this phase of the development in this document can be traced to the work and choices articulated in the appendix within

draft-ietf-tsvwg-intserv-multiple-tspec

From there, another team developed

draft-lefaucheur-tsvwg-rsvp-multiple-preemption

Then Lou Berger (yeah, blame him!) came up with the bright idea of

combining the two efforts in such a way that one can take a complete Object or element, and replace the primary instance of that within an RSVP message for whatever reason (perhaps the message would be rejected if that piece was not replaced with more reasonable demands on the network; who knows). Anyway, that's how we got here. That's the story and I'm sticking to it... :-p