Internet Draft
Expiration: July 1999
File: draft-ietf-rap-rsvp-ext-02.txt

Shai Herzog
IPHighway

# RSVP Extensions for Policy Control

January 22, 1999

## Status of this Memo



## Abstract

This memo presents a set of extensions for supporting generic policy based admission control in RSVP. It should be perceived as an extension to the RSVP functional specifications [RSVP]

These extensions include the standard format of POLICY_DATA objects, and a description of RSVP's handling of policy events.

This document does not advocate particular policy control mechanisms; however, a Router/Server Policy Protocol description for these extensions can be found in [RAP, COPS, COPS-RSVP].

Table of Contents

## 1. Introduction

RSVP, by definition, discriminates between users, by providing some users with better service at the expense of others. Therefore, it is reasonable to expect that RSVP be accompanied by mechanisms for controlling and enforcing access and usage policies.  Historically, when RSVP Ver. 1 was developed, the knowledge and understanding of policy issues was in its infancy. As a result, Ver. 1 of the RSVP Functional Specifications [RSVP] left a place holder for policy support in the form of POLICY_DATA objects. However, it deliberately refrained from specifying mechanisms, message formats, or providing insight into how policy enforcement should be carried out. This document is intended to fill in this void.

The current RSVP Functional Specification describes the interface to admission (traffic) control that is based "only" on resource availability. In this document we describe a set of extensions to RSVP for supporting policy based admission control as well. The scope of this document is limited to these extensions and does not advocate specific architectures for policy based controls.

For the purpose of this document we do not differentiate between Policy Decision Point (PDP) and Local Decision Point (LDPs) as described in [RAP]. The term PDP should be assumed to include LDP as well.

## 2. Policy Data Object Format

The following replaces section A.13 in [RSVP].

POLICY_DATA objects are carried by RSVP messages and contain policy information. All policy-capable nodes (at any location in the network) can generate, modify, or remove policy objects, even when senders or receivers do not provide, and may not even be aware of policy data objects.

The exchange of POLICY_DATA objects between policy-capable nodes along the data path, supports the generation of consistent end-to-end policies. Furthermore, such policies can be successfully deployed across multiple administrative domains when border nodes manipulate and translate POLICY_DATA objects according to established sets of bilateral agreements.

## 2.1. Base Format

POLICY_DATA class=14

o   Type 1 POLICY_DATA object: Class=14, C-Type=1

```
+-------------+-------------+-------------+-------------+
|  Length                   | POLICY_DATA |      1      |
+---------------------------+-------------+-------------+
|  Data Offset              | 0 (reserved)              |
+---------------------------+-------------+-------------+
|                                                       |
// Option List                                         //
|                                                       |
+-------------------------------------------------------+
|                                                       |
// Policy Element List                                 //
|                                                       |
+-------------------------------------------------------+
```

Data Offset: 16 bits

The offset in bytes of the data portion (from the first byte of the object header).

Reserved: 16 bits

Always 0.

Option List: Variable length

The list of options and their usage is defined in Section 2.2.

Policy Element List: Variable length

The contents of policy elements is opaque to RSVP. See more details in Section 2.3.

## 2.2. Options

This section describes a set of options that may appear in POLICY_DATA objects. All policy options appear as RSVP objects; some use their valid original format while others appear as NULL objects.

### 2.2.1. Native RSVP Options

The following objects retain the same format specified in [RSVP] however, they gain different semantics when used inside POLICY_DATA objects.

FILTER_SPEC object (list) or SCOPE object

The set of senders associated with the POLICY_DATA object. If none is provided, the policy information is assumed to be associated with all the flows of the session. These two types of objects are mutually exclusive, and cannot be mixed.

This option is only useful for WF or SE reservation styles, where merged reservations may have originally been intended for different subsets of senders. It can also be used to prevent  policy loops  in a manner similar to the usage of RSVP s SCOPE object. Using this option may have significant impact on scaling and size of POLICY_DATA objects and therefore should be taken with care.

Originating RSVP_HOP

The RSVP_HOP object identifies the neighbor/peer policy-capable node that constructed the policy object. When policy is enforced at border nodes, peer policy nodes may be several RSVP hops away from each other and the originating RSVP_HOP is the basis for the mechanism that allows them to recognize each other and communicate safely and directly.

If no RSVP_HOP object is present, the policy data is implicitly assumed to have been constructed by the RSVP_HOP indicated in the RSVP message itself (i.e., the neighboring RSVP node is policy-capable).

Destination RSVP_HOP

A second RSVP_HOP object may follow the originating RSVP_HOP object. This second RSVP_HOP identifies the destination policy node. This is used to ensure the POLICY_DATA object is delivered to targeted policy nodes. It may be used to emulate unicast delivery in multicast Path messages. It may also help prevent using a policy object in other parts of the network (replay attack).

On the receiving side, a policy node should ignore any POLCY_DATA that includes a destination RSVP_HOP that doesn t match its own IP address.

INTEGRITY Object

The INTEGRITY object provides guarantees that the object was not compromised. It follows the rules from [MD5], and is calculated over the POLICY_DATA object, the SESSION object, and the message type field

(byte, padded with zero to 32 bit) as if they formed one continuous in-order message.  This concatenation is designed to prevent copy and replay attacks of POLICY_DATA objects from other sessions, flows, message types or even other network locations.

#### 2.2.2. Other Options

All options that do not use a valid RSVP object format, should use the NULL RSVP object format with different CType values. This document defines only one such option, however, several other may be considered in future versions.  (e.g., Fragmentation, NoChange, etc.).

o    Policy Refresh Period (PRP)

The Policy Refresh Period (PRP) option is used slow down policy refresh frequency for policies that have looser timing constraints compared with RSVP. If the PRP option is present, policy refreshes can be withheld as long as at least one refresh is sent before the policy refresh timer expires (PRP must be bigger than R).

```
+-------------+-------------+-------------+-------------+
|            8              |    NULL     |     1       |
+-------------+-------------+-------------+-------------+
|         Policy Refresh Period (PRP) (in seconds)      |
+-------------+-------------+---------------------------+
```

It is recommended that this infrequent policy refresh would be piggybacked with normal RSVP refreshes. Given an RSVP refresh R, the policy must be refreshed at least once in N RSVP refreshes, where N=Floor(PRP/R) and the Floor function provides the integer portion of the result.

In effect, state cleanup rules apply specifically to the POLICY_DATA object as if the RSVP refresh period was N*R.

Any RSVP update must include the full policy information. For example, a policy being refreshed at time T, T+N, T+2N,... may encounter a route change detected at T+X such that T < T+X < T+N. The update event would force an immediate update of the policy and change its refresh times to T+X, T+X+N, T+X+2N,...

When network nodes restart, it is possible that an RSVP message in between policy refreshes would be rejected since it arrives to a node that did not receive the original POLICY_DATA object.  This error situation would clear with the next periodic policy refresh or by an update triggered by ResvErr or PathErr messages.

This option is especially useful to combine strong (high overhead) and weak (low overhead) authentication certificates. In such schemes the

weak certificate supports admitting a reservation only for a limited time, after which the strong certificate is required.

This approach may reduce the overhead of POLICY_DATA processing. Strong certificates could be transmitted less frequently, while weak certificates could be included in every RSVP refresh.

### 2.3. Policy Elements

The content of policy elements is opaque to RSVP; their internal format is understood by policy peers e.g. an RSVP Local Decision Point (LDP) or a Policy Decision Point (PDP) [RAP]. A registry of policy element codepoints and their meaning is maintained by [IANA-CONSIDERATIONS] (also see Section 4).

Policy Elements have the following format:

```
+-------------+-------------+-------------+-------------+
|  Length                   |   P-Type                  |
+---------------------------+---------------------------+
|                                                       |
// Policy information  (Opaque to RSVP)               //
|                                                       |
+-------------------------------------------------------+
```

## 3. Processing Rules

These sections describe the minimal required policy processing rules for RSVP.

### 3.1. Basic Signaling

It is generally agreed that policy control should only be enforced for Path, Resv, PathErr, and ResvErr. PathTear and ResvTear and assumed not to require policy control based on two assumptions: First, that Integrity verification [MD5] guarantee that the Tear is received from the same node that sent the installed reservation, and second, that it is functionally equivalent to that node holding-off refreshes for this reservation.

### 3.2. Default Handling

It is generally assumed that policy enforcement (at least in its initial stages) is likely to concentrate on border nodes between autonomous systems. Consequently, policy objects transmitted at one edge of an autonomous cloud may traverse intermediate policy ignorant RSVP nodes (PINs). A PIN is required at a minimum to forward the received POLICY_DATA objects in the appropriate outgoing messages according to the following rules:

- o POLICY_DATA objects are to be forwarded as is, without any modifications.

o    Multicast merging (splitting) nodes:

In the upstream direction:

When multiple POLICY_DATA objects arrive from downstream, the RSVP node should concatenate all of them and forward them with the outgoing (upstream) message.

On the downstream direction:

When a single incoming POLICY_DATA object arrives from upstream, it should be forwarded (copied) to all downstream branches of the multicast tree.

The same rules apply to unrecognized policies (sub-objects) within the POLICY_DATA object. However, since this can only occur in a policy-capable node, it is the responsibility of the PDP and not RSVP.

### 3.3. Error Signaling

Policy errors are reported by either ResvErr or PathErr messages with a policy failure error code in the ERROR_SPEC object. Policy error message must include a POLICY_DATA object; the object contains details of the error type and reason in a P-Type specific format.

If a multicast reservation fails due to policy reasons, RSVP should not attempt to discover which reservation caused the failure (as it would do for Blockade State). Instead, it should attempt to deliver the policy ResvErr to ALL downstream hops, and have the PDP (or LDP) decide where messages should be sent. This mechanism allows the PDP to limit the error distribution by deciding which "culprit" next-hops should be informed. It also allows the PDP to prevent further distribution of ResvErr or PathErr messages by performing local repair (e.g. substituting the failed POLICY_DATA object with a different one).

Error codes are described in Appendix A.

## 4. IANA Considerations

RSVP Policy Elements

Following the policies outlined in [IANA-CONSIDERATIONS],numbers 0-49151 are allocated as standard policy elements by IETF Consensus action, numbers in the range 49152-53247 are allocated as vendor specific (one per vendor) by First Come First Serve, and numbers 53248-65535 are reserved for private use and are not assigned by IANA.

## 5. References

[RAP]  Yavatkar, R., et al., "A Framework for Policy Based Admission Control",IETF <draft-ietf-rap-framework-02.txt>, Jan., 1999.

[COPS] Boyle, J., Cohen, R., Durham, D., Herzog, S., Raja,n R., Sastry, A., "The COPS (Common Open Policy Service) Protocol", IETF <draft-ietf-rap-cops-05.txt>, Jan. 1999.

[RSVP] Braden, R. ed., "Resource ReSerVation Protocol (RSVP) - Functional Specification.", IETF RFC 2205, Proposed Standard, Sep. 1997.

[MD5]  Baker, F., Linden B., Talwar, M.  RSVP Cryptographic Authentication" Internet-Draft, <draft-ietf-rsvp-md5-07.txt>, Nov. 1998.

[IANA-CONSIDERATIONS]  Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, October 1998.

## 6. Acknowledgments

This document incorporates inputs from Lou Berger, Bob Braden, Deborah Estrin, Roch Guerin, Timothy O'Malley, Dimitrios Pendarakis, Raju Rajan, Scott Shenker, Andrew Smith, Raj Yavatkar, and many others.

## 7. Author Information

Shai Herzog, IPHighway
Parker Plaza, Suite 1500
400 Kelby St.
Fort-Lee, NJ 07024
(201) 585-0800
herzog@iphighway.com

## A. Appendix: Policy Error Codes

This Appendix expends the list of error codes described in Appendix B of [RSVP].

Note that Policy Element specific errors are reported as described in Section 3.3 and cannot be reported through RSVP (using this mechanism). However, this mechanism provides a simple, less secure mechanism for reporting generic policy errors. Most likely the two would be used in concert such that a generic error code is provided by RSVP, while Policy Element specific errors are encapsulated in a return POLICY_DATA object (as in Section 3.3).

ERROR_SPEC class = 6

Error Code = 02: Policy Control failure

Error Value: 16 bit

```
0 = ERR_INFO    : Information reporting
1 = ERR_WARN    : Warning
2 = ERR_UNKNOWN : Reason unknown
3 = ERR_REJECT  : Generic Policy Rejection
4 = ERR_EXCEED  : Quota or Accounting violation
5 = ERR_PREEMPT : Flow was preempted
6 = ERR_EXPIRED : Previously installed policy expired (not refreshed)
7 = ERR_REPLACED: Previous policy data was replaced & caused rejection
8 = ERR_MERGE   : Policies could not be merged (multicast)
9 = ERR_PDP     : PDP down or non functioning
10= ERR_SERVER  : Third Party Server (e.g., Kerberos) unavailable
11= ERR_PD_SYNTX: POLICY_DATA object has bad syntax
12= ERR_PD_INTGR: POLICY_DATA object failed Integrity Check
13= ERR_PE_BAD  : POLICY_ELEMENT object has bad syntax
14= ERR_PD_MISS : Mandatory PE Missing (Empty PE is in the PD object)
15= ERR_NO_RSC  : PEP Out of resources to handle policies.
16= ERR_RSVP    : PDP encountered bad RSVP objects or syntax
17= ERR_SERVICE : Service type was rejected
18= ERR_STYLE   : Reservation Style was rejected
19= ERR_FL_SPEC : FlowSpec was rejected (too large)
```

Values between 2^15 and 2^16-1 can be used for site and/or vendor error values.