Internet Draft
Expiration: Feb 1999
File: draft-ietf-rap-cops-rsvp-00.txt

Jim Boyle
L3
Ron Cohen
Cisco
David Durham
Intel
Shai Herzog
IPHighway
Raju Rajan
IBM
Arun Sastry
Cisco

COPS usage for RSVP

Last Updated: August 19, 1998

Status of this Memo

Abstract

This document describes usage directives for supporting COPS policy services in RSVP environments.

1. Introduction

The Common Open Policy Service (COPS) protocol is a query response protocol used to exchange policy information between a network policy server and a set of clients [COPS]. COPS is being developed within the RSVP Admission Policy Working Group (RAP WG) of the IETF, primarily for use as a mechanism for providing policy-based admission control over requests for network resources [RAP].

This document is based on and assumes prior knowledge of RAP framework [RAP] and the basic COPS [COPS] protocol. It provides specific usage directives for using COPS in outsourcing policy control decisions by RSVP clients (PEPs) to policy servers (PDPs).

Given the COPS protocol design, client specific functionality is mainly limited to interoperability usage guidelines as well as client specific examples.

2. RSVP values for COPS objects

The format and usage of several COPS objects is affected when used for client type RSVP. This section describes these objects and the usage.

2.1. Context Object (Context)

The semantics of the Context object for RSVP is as follows:

R-Type (Request Type Flag)

0x01 = Incoming-Message request
The arrival of an incoming RSVP message

Allows processing of incoming policy information as well as the decision whether to accept an incoming message. If It is rejected, the message is treated as if it never Arrived.

0x02 = Resource-Allocation request
Applies only for Resv messages.

The decision whether to admit a reservation and commit local resources to it is performed for the merge of all reservations that arrived on a particular interface (potentially from several Previous Hops).

0x04 = Outgoing-Message request
The forwarding of an outgoing RSVP message.

The Decision whether to allow the forwarding of an outgoing

RSVP message as well as providing the relevant outgoing policy information.

M-Type (Message Type)

The M-Type field in the Context Object may have one of the Following values that correspond to supported RSVP messages In COPS:

1 = Path
2 = Resv
3 = PathErr
4 = ResvErr

Note: The PathTear, ResvTear, and the Resv Confirm message types are not supported.

2.2. Client Specific Information (ClientSI)

All objects contained within RSVP messaging are encapsulated inside the Client Specific Information Object without alteration. The multiple RSVP objects are simply contained within a single Signalled Client Specific Information Object (Signaled ClientSI) exchanged between the PEP and remote PDP.

To prevent ambiguity, the number of object instances appearing in the ClientSI are restricted by the rules native to RSVP. For example, it is forbidden to include two different FlowSpec objects in one ClientSI encapsulation, while perfectly legal to include multiple FilterSpecs for a WF or SE reservation.

For applicability example, see Section 3.6.

2.3. Decision Object (Decision)

COPS allows PDP to control RSVP's response to messages. Beyond traditional accept/deny, PDPs may use the Trigger Error flag to allow a request yet trigger a warning at the same time. To allow

resource allocation yet deny forwarding of a message, etc.

Replacement Data

The Replacement object may contain multiple RSVP objects to be replaced (from the original RSVP request). Typical replacement is performed on the "Forward Outgoing" request (for instance, replacing outgoing Policy Data), but is not limited to this context flag. Another example, for controlling resources across a trusted zone (with PIN nodes) the RSVP FlowSpec object may need to be replaced.

Currently, RSVP clients are only required to allow replacement of two objects: Policy Data and FlowSpec.

Client Specific decision Object

In support of the verification integrity of incoming RSVP messages, the COPS protocol may optionally return a security key in the Client Specific Decision Data object (C-Type = 4) useful for future integrity checks by the PEP. Refer to the document on User Identity Representation for RSVP [UserID] for details on the format and application of this security key when supported by the PEP.

3. Operation of COPS for Policy Control Over RSVP

3.1. RSVP flows

Policy Control is performed per RSVP flow. An RSVP flow corresponds to an atomic unit of reservation as identified by RSVP (TC reservation). It should be noted that RSVP allows multiple flows to be packed (which is different from merged) into a single FF Resv message. To support such messages a separate COPS request must be issued for each of the packed flows as if they were individual RSVP messages.

3.2. Expected Associations for RSVP Requests

RSVP signaling requires the participation of both senders and receivers. RSVP processing rules define what is the subset of the Path state that matches each Resv state. In the common unicast case, the RSVP session includes one Path state and one Resv state. In multicast cases the correspondence might be many to many. Since the decision to admit a reservation for a session may depend on

information carried both in Path and Resv messages, we term the Path States that match with a single Resv state as its associated states. It is assumed that the PDP is capable of determining these associations based on the RSVP message processing rules given the RSVP objects expressed in the COPS Client Specific Information Object.

3.3. RSVP's Capacity Admission Control: Commit and Delete

In RSVP, the admission of a new reservation requires both an administrative approval (policy control) and capacity admission control. Once local admission control accepts the reservation, the PEP notifies the remote PDP by sending a report message specifying the Commit type. The Commit type report message is to be used to signify when billing should effectively begin, and performing heavier operations (e.g., debiting a credit card) is permissible.

If instead a reservation approved by the PDP fails admission due to lack of resources, the PEP must resort to its previous state (previously installed reservation). If none was previously installed, the PEP should issue a delete. Otherwise, it should issue a report no-commit and then send a request update for the previously allocated reservation state.

3.4. Policy Control Over Path and Resv Tear

Path and Resv Tear messages are not controlled by this policy architecture. This relies on two assumptions: First, that MD-5 authentication verifies that the Tear is received from the same node that sent the initial reservation, and second, that it is functionally equivalent to that node holding-off refreshes for this reservation. When a Resv or Path Tear is received at the PEP, all affected states installed on the PDP should either be deleted or updated by the PEP.

3.5. PEP Caching COPS Decisions

Because COPS is a stateful protocol, refreshes for RSVP Path and Resv messages need not be constantly sent to the remote PDP. Once a decision has been returned for a request, the PEP can cache that decision and apply it to future refreshes. The PEP is only responsible for updating a request state if there is a change detected in the corresponding Resv or Path message.

If the connection is lost between the PEP and the PDP, the cached RSVP state may be retained for the RSVP timeout interval. If no connection can be reestablished with the PDP or a backup PDP, the RSVP PEP is expected to default back to using its LDP. Additionally, the LDP is to be used for the admission control of any new RSVP messages that may have arrived while connectivity was lost. If any such messages were admitted by the LDP, the PEP is expected to issue requests to the PDP for them once a connection is reestablished and a COPS session for RSVP is opened.

3.6. Using Multiple Context Flags in a single query

RSVP is a store-and-forward control protocol where messages are processed in three distinctive steps (input, output and resource allocation). Each step requires a separate policy decision as indicated by context flags (see Section 2.1). In many cases, setting multiple context flags can serve as a mean for bundling two of three operations together in one request (for instance, validating both an incoming message as well as allocating resources for it).

The following rules apply when multiple Context flags are set:

a. Multiple context flags can be set simultaneously when their merge doesn't create ambiguity in ClientSI interpretation. Two context flags become mutually exclusive for a specific REQ or DEC message when identical ClientSI objects carry different values for each of them.

b. The DEC message contains a context object that correspond to all or a subset of the context flags set in original REQ. A DEC is assumed to be complete (as far as reuse of handles) only when all flags have been replied to (set in the following DEC message).

c. The PEP may act based on partial (context subset) decision and is not required to wait for the others, although it may. Consistency is an issue for the PDP to worry about.

3.7. Trusted zones and secure policy tunneling

Security for RSVP messages is provided by inter-router MD5 authentication [MD5], assuming a chain-of-trust model.
A possible deployment scenario calls for PEPs to be deployed at the

network edge (boundary nodes) while PINs are deployed in the core of the network (backbone). In this case, MD5 trust (authentication) must be established between boundary (non-neighboring) PEPs, which is achieved through internal signing of the Policy Data object. [RSVP-EXT].

4. Illustrative Examples, Using COPS for RSVP

This section details both typical unicast and multicast scenarios.

4.1. Unicast Flow Example

This section details the steps in using COPS for controlling a Unicast RSVP flow. It details the contents of the COPS messages with respect to the following figure.

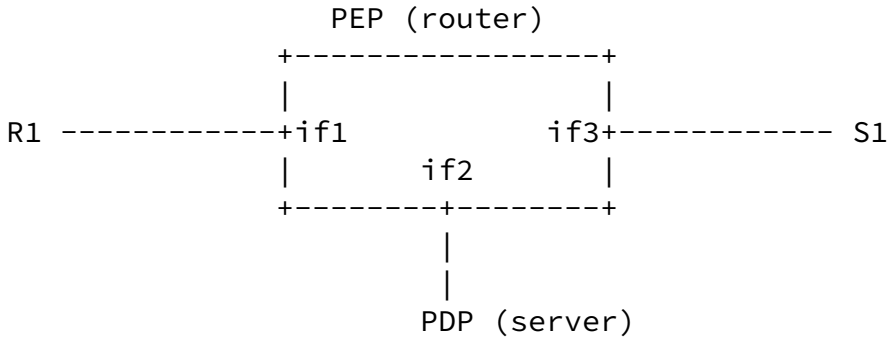


figure 1: Unicast Example: a single router view

The PEP router has three interfaces (1,2,3). Sender S1 sends to receiver R1.

A Path message arrives from S1:

```
PEP --> PDP   REQ := <Handle A><Context in&out, Path>
                     <In-Interface if3> <Out-Interface if1>
                     <ClientSI: all objects in Path message>

PDP --> PEP   DEC := <Handle A>><Context in&out, Path>
                     <Decision accept>
```

A Resv message arrives from R1:

```
        PEP --> PDP   REQ := <Handle B><Context in&merge&out, Resv>
                             <In-Interface if1> <Out-Interface if3>
                             <ClientSI: all objects in Resv message>

        PDP --> PEP   DEC := <Handle B>
                             <Context in&merge&out, Resv>
                             <Decisions: accept, Priority=7,
                              Replace: POLICY.DATA1>

        PEP --> PDP   RPT := <Handle B>
                             <Report type: commit>
```

Time Passes, the PDP changes its decision:

```
        PDP --> PEP   DEC := <Handle B>
                             <Context in&merge&out, Resv>
                             <Decisions: accept, Priority=3,
                              Replace: POLICY.DATA2>
```

Because the priority is too low, the PEP preempts the flow:

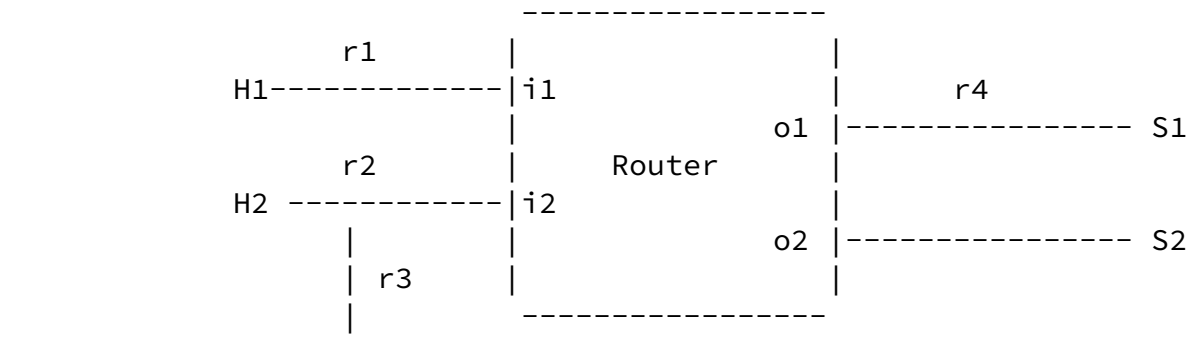
```
        PEP --> PDP   DRQ := <Handle B>
                             <Reason Code: Preempted>
```

Time Passes, the sender S1 ceases to send Path messages:

```
        PEP --> PDP   DRQ := <Handle A>
                             <Reason: Timeout>
```

4.2. Shared Multicast Flows

This section details the steps in using COPS for controlling a multicast RSVP flow. It details the contents of the COPS messages with respect to the following figure.



H3

figure 1: 2 senders and 3 receivers

Figure 1 shows an RSVP router which has two senders and three receivers for the same multicast session. Interface i2 is connected to a shared media. H1, H2, and H3 are all receivers that send the corresponding reservations r1, r2, and r3 for the flows for senders S1 & S2.

First detailed is the request message content for a Path sent by sender S1, assuming that both receivers have already joined the multicast session, but haven't sent a Resv message as yet. Assume sender S2 has not yet sent a path message. The Path message arrives on interface o1:

```
PEP -----> PDP    REQ   := <handle A><context in, Path>
                           <in-interface o1><client info: all
                           objects in Path message>
PDP -----> PEP    DEC   := <handle A><context in, Path>
                           <Decision accept>
```

Here the PDP decides to allow the Path message. Next, the Router consults its forwarding table, and finds two outgoing interfaces, i1 and i2, for the path. The exchange below is for interface i1, another exchange would likewise be completed for i2 using the new handle B2.

```
PEP -----> PDP    REQ   := <handle B1><context out, Path>
                           <out-interface i1><client info: all
                           objects in outgoing Path message>
PDP -----> PEP    DEC   := <handle B1><Decision forward>
                           <context out, Path>
                           <Decision replacement object:
                            policy object>
```

Here, the PDP decided to allow the forwarding of the Path message via interface i1, and determined the appropriate policy objects for the message going out on this interface.

Next, the receiver r2 sends a Resv message of WF style. The Resv arrives on interface i2. Here the PEP queries the PDP which decides to accept this reservation with priority 5 as shown below.

```
    PEP -----> PDP    REQ   := <handle C><context in, Resv>
                               <in-interface i2><client info: all
                               objects in Resv message>
    PDP -----> PEP    DEC   := <handle C><Context in, Resv>
                               <Decision accept>
```

This assumes the PEP is not itself capable of merging priority information, and, thus, must make another query for the incoming interface merge.

```
    PEP -----> PDP    REQ   := <handle D><context merge, Resv>
                               <in-interface i2><client info: all
                               objects in merged Resv message>
    PDP -----> PEP    DEC   := <handle D><context merge, Resv>
                               <Decision Priority: 5>
```

After PEP successfully admitted the reservation it sends a report message that signals to the PDP that it can start an accounting log for this reservation.

```
    PEP -----> PDP    RPT   := <handle D>
                               <commit>
```

The reservation r2 needs to be sent upstream towards sender S1 out interface o1. An outgoing Resv request is made which carries the associated handle of the Path message for which this Resv is being forwarded.

```
    PEP -----> PDP    REQ   := <handle E><context out,Resv>
                               <out-interface o1><client info: all
                               objects in outgoing Resv message>
    PDP -----> PEP    DEC   := <handle E><Context out, Resv>
                               <Decision forward><Decision
                               replacement object: policy object>
```

Next, receiver H3 sends the Resv message r3. The PEP sends an incoming request for handle F and the PDP decides to accept the Resv (as before). The new reservation also requires the PEP to update the merged request (handle D) due to the modified flowspec. The PDP now gives this request priority 7. If accepted by local admission control, a report is again sent.

```
    PEP -----> PDP    REQ   := <handle D><context merge, Resv>
                               <in-interface i2><client info: all
                               objects in merged Resv message w/
                               new merged FLOWSPEC>
```

```
PDP -----> PEP    DEC   := <handle D><Context merge, Resv>
                           <Decision priority 7>
PEP -----> PDP    RPT   := <handle D>
                           <commit>
```

Now the outgoing request for handle E is reissued for the merged (R2 & R3) outgoing Resv to be sent towards sender S1 due to a modified flowspec.

```
PEP -----> PDP    REQ   := <handle E><context out,Resv>
                           <out-interface o1><client info: all
                           objects in outgoing Resv message w/
                           new merged FLOWSPEC>
PDP -----> PEP    DEC   := <handle E><Context out, Resv>
                           <Decision forward><Decision
                           replacement object: policy object>
```

When S2 joins the session by sending a Path message, incoming and outgoing Path requests are issued for the new Path. The two incoming Resv requests may then be reissued for handle C and handle E if there is a change in their shared sender filter list (e.g. if this was a SE filter example) specifying the new sender. A new outgoing Resv request would then be issued for the Resv to be sent to s2 out interface o2.

5. References

[RSVP-EXT] Herzog, S. "RSVP Extensions for Policy Control", Internet-Draft, draft-ietf-rap-rsvp-ext-00.txt, Apr. 1998.

[UserID] Yadav, S., Pabbati, R., Ford, P., Herzog, S., "User Identity Representation for RSVP", Internet-Draft, draft-ietf-rap-user-identity-00.txt, March 1998.

[RAP] Yavatkar, R., et al., "A Framework for Policy Based Admission Control",IETF <draft-ietf-rap-framework-00.txt>, November, 1997.

[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-02.txt>, Aug. 1998.

[RSVP] Braden, R., Zhang, L., Berson, S., Herzog, S., and Jamin, S., "Resource Reservation Protocol (RSVP) Version 1 Functional Specification", IETF RFC 2205, Proposed Standard, September 1997.

6. Author Information and Acknowledgments

Special thanks to Timothy O'Malley our WG Chair, Raj Yavatkar, Russell Fenger, Fred Baker, Laura Cunningham, Roch Guerin, Ping Pan, and Dimitrios Pendarakis for their valuable contributions.

Jim Boyle
Level 3 Communications
1450 Infinite Drive13
Louisville, CO 80027
303.926.3100
email: jboyle@l3.net

Ron Cohen
CISCO Systems
Hasadna St.
Ra'anana 43650 Israel
972.9.7462020
ronc@cisco.com

David Durham
Intel
2111 NE 25th Avenue
Hillsboro, OR 97124
503.264.6232
David_Durham@mail.intel.com

Raju Rajan
IBM T.J. Watson Research Cntr
P.O. Box 704
Yorktown Heights, NY 10598
914.784.7260
raju@watson.ibm.com

Shai Herzog
IPHighway
2055 Gateway Pl., Suite 400
San Jose, CA 95110
408.390.3045
herzog@iphighway.com

Arun Sastry
Cisco Systems
506210 W Tasman Drive
San Jose, CA 95134
408.526.7685
asastry@cisco.com

Table of Contents