Network Working Group
Internet Draft
draft-sgai-cops-provisioning-00.txt
Expiration Date: August 1999

Francis Reichmeyer
Kwok Ho Chan
Nortel Networks, Inc.
David Durham
Raj Yavatkar
Intel
Silvano Gai
Keith McCloghrie
Cisco Systems, Inc.
Shai Herzog
IPHighway
Andrew Smith
Extreme Networks
February 1999

COPS Usage for Policy Provisioning

Status of this Memo

Copyright Notice

Abstract

There is a clear need for a standard way to provision policies to network devices. These policies may be related to QoS (Quality of Service), Security, VPNs (Virtual Private Networks), etc.

The IETF RSVP Admission Policy (RAP) WG has defined the COPS (Common Open Policy Service) protocol [COPS] and a scalable policy control model for RSVP [RSVP].

This document describes a new client type ("Provisioning") for the Common Open Policy Service (COPS) protocol to support policy provisioning. This new client type is independent of the type of policy and it is based on the concept of PIBs (Policy Information Bases [PIB].

The example of provisioning used in this document is QoS Policy Provisioning in a Differentiated Services (DiffServ) environment.

Table of contents

1. Terminology

- ClientSI: Client Specific Information Object.
- COPS (Common Open Policy Service): client/server model for supporting policy control [COPS].
- Object: this term is used in the same sense as in COPS specification. A COPS object is identified by its C-Num and C-Type, a ClientSI object by its S-Num, S-Type.
- PDP (Policy Decision Point): a network entity where policy decisions are made.
- PEP (Policy Enforcement Point): network device where policy decisions are enforced.
- Policy Rule: policy information specified by the PDP to be enforced at the PEP.
- PRC (Policy Rule Class): a type of policy rule data item. In object oriented terminology this is equivalent to a class. A PRC defines a vector of attributes. Each attribute has a syntax type.
- PRI (Policy Rule Instance): an instance of a PRC. Potentially there are multiple instances of the same PRC. The value of a PRI consist of a vector of values, one value for each attribute in the PRC's vector of attributes.
- PII (Policy Instance Identifier): one or more of the PRC attributes the values of which are used as part of the identification of a PRI.
- PIB (Policy Information Base): policy objects are accessed via a virtual information store, termed the Policy Information Base or PIB [PIB]. Objects in the PIB are defined using a subset of Abstract Syntax Notation One (ASN.1) [ASN1].
- PRID (Policy Rule IDentifier): the name which identifies a particular PRI or PRC. It has a hierarchical structure of the form

1.3.4.2.7, where the first part identifies the PRC (i.e., 1.3.4) and the last part is the value of the PII (Policy Instance Identifier), which identifies the instance (i.e. 2.7). The PII is null in the case of a PRC. PRIDs are represented as a BER encoded OIDs (Object Identifiers).

- BPD: BER (ASN.1 [ASN1] Basic Encoding Rule [BER]) encoded Policy Instance Data.

2. 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].

The underlying assumption in the RAP framework is that applications or end systems use the RSVP [RSVP] signaling protocol to communicate Integrated Services (IntServ) reservation requests to the network nodes along the path of a flow. These reservation requests carry necessary flow specifications and requests for a flow to receive one of the defined Integrated Services, Controlled Load or Guaranteed. In the IntServ model, the RSVP messages themselves contain all the necessary information needed at the networking device to classify and service the flow [RSVP]. This information includes the session identifier (source and destination addresses, port numbers, and transmission protocol), flowspec token bucket parameters, and requested service.

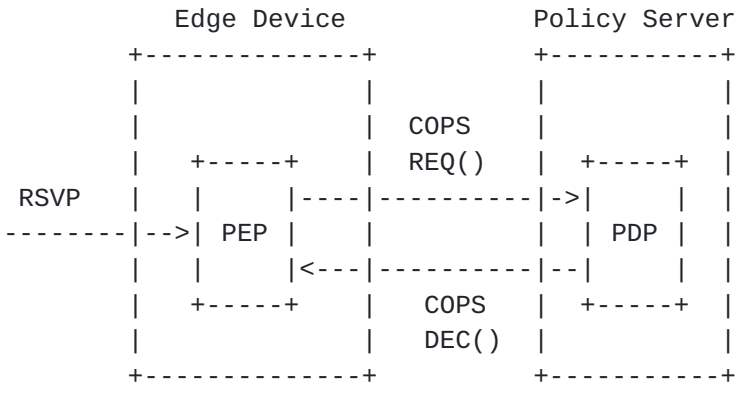


Figure 1: COPS with RSVP/IntServ

As shown in Figure 1, the network device contacts a Policy Decision Point (PDP) to make the policy-based admission control decision. The PDP is simply required to return a Decision, such as "accept" and the network device acts as a Policy Enforcement Point (PEP) and uses the session information and IntServ service parameters to classify and service the packets belonging to the flow.

Providing policy services in a DiffServ environment requires some different assumptions about the admission control mechanisms used in the network. First, there might be no explicit dynamic signaling from sources of traffic requesting a particular service, as in the case of an IntServ network. Network resources are provisioned based on static SLAs (Service Level Agreements) at network boundaries. Second, where requests for allocation of resources to differentiated services are used, they may arrive at the PDP from network entities other than the PEP. Examples of such sources include attached users requesting network services via a web interface into a central management application, or H.323 gatekeeper requesting resources on behalf of a user for a video conferencing application, as shown in Figure 2.

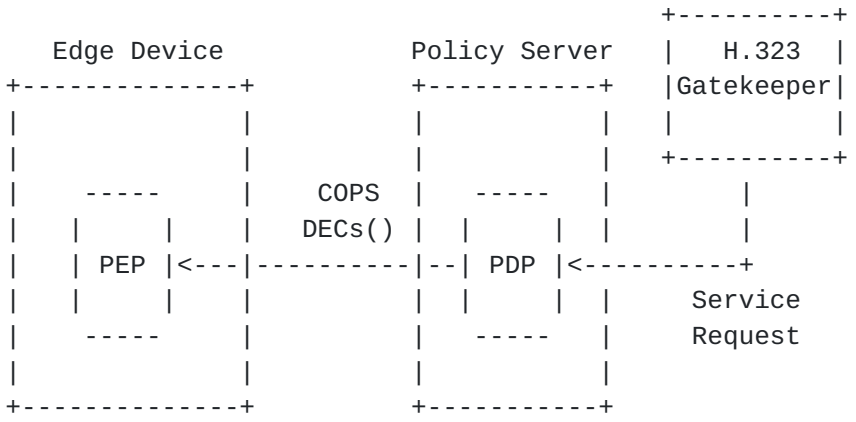


Figure 2: COPS Example with DiffServ

Requests of this sort still require some policy decision to be made to ensure the requesting user/application has permission to use the requested services and that the resources are available. Once the decision is made, the PDP must configure one or more PEPs to allocate necessary resources for services requested. In addition, the PDP may also pass to the PEP provisioning decisions about resources related to flows of a more static nature, such as long-term SLAs established across boundaries of adjacent ISP networks.

In summary, the interaction between the PDP and PEP is different in at least two respects from that in the case of the IntServ environment. First, the resource provisioning requests may originate at places other than a PEP. Second, once the PDP makes a policy decision to allocate resources for a service class or a flow aggregate, it must pass sufficient information (such as packet classification filters, traffic shaper parameters) in the decision message to the PEPs so that PEPs can enforce policy decisions.

This draft describes a new client type ("Provisioning") for COPS to support policy provisioning. This new client type is independent of the type of policy (QoS, VPNs, Security, etc.) and it is based on the concept of PIBs (Policy Information Bases [PIB]).

Adding a new client type to COPS is much easier than designing a new protocol and allows us to reuse all the COPS code. Moreover, for QoS Provisioning it has the additional advantage to adopt a single protocol for RSVP and provisioned QoS. We have taken another important decision to simplify the implementation of Provisioning, i.e., to reuse the BER [BER] encoding and therefore part of the code already available for SNMP.

2.1 Basic Model

Figure 3 shows a sample network configuration for a DiffServ environment.

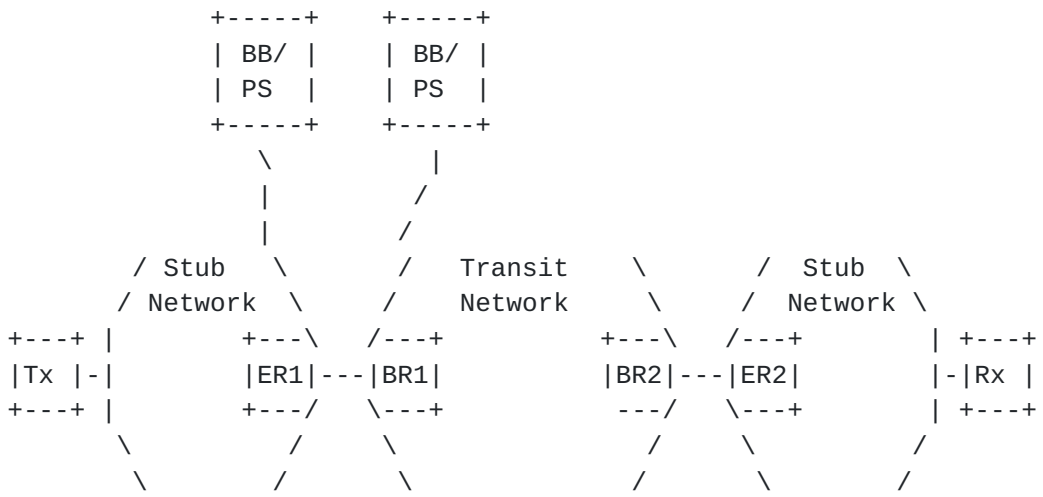


Figure 3: Sample DiffServ Network Configuration

Edge routers and boundary routers are located at the boundary of DiffServ domains as described in [RFC 2475]. The BB/PS is responsible for admission control functions and resource provisioning.

In the COPS model, the PDP is part of the bandwidth broker/policy server that manages policy information and resources within a DiffServ domain. Both edge routers and boundary routers act as PEPs and communicate with BB/PS using COPS for exchange of policy information. The internal organization of the bandwidth broker functionality and policy functionality may vary and the policy server and BB may be separate entities. In that case, either the BB or the PS may communicate with the edge devices. The BB, upon receiving COPS messages from the PEP, would consult the policy server to make its final admission control decision. Similarly, if the PS receives COPS messages directly from PEP, the PS would consult the BB to verify available resources before making a final admission control decision.

To allow for use of COPS for policy provisioning and to distinguish this usage from other uses of COPS, we have added a new client type to COPS (client type = Provisioning client). It is possible for an edge device to contain both a COPS-PR (COPS-Provisioning) and a COPS-RSVP client. Each COPS clients can communicate with different PDPs, or they can connect to the same PDP which supports both client types, as shown in Figure 4.

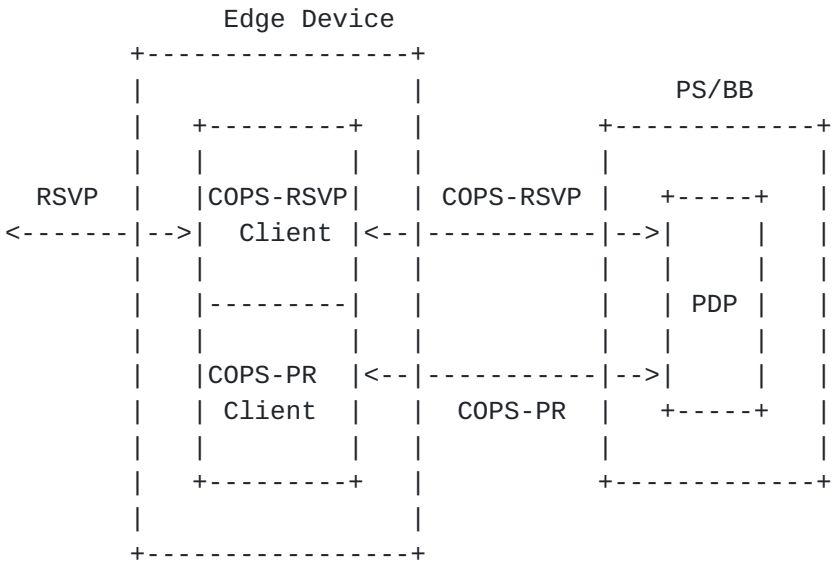


Figure 4: COPS-PR and RSVP Clients in Same Edge Device

Allowing multiple COPS client types to co-exist in a single PEP means that the same PDP can coordinate policy decisions in an environment where, say, both RSVP/IntServ and DiffServ QoS mechanisms need to be managed. For example, in a stub network that uses IntServ with RSVP signaling internally and is connected to a DiffServ transit network externally. In this case, the edge device that connects the stub network to the transit network may require policy decisions from the same PDP for both RSVP requests as well as for policy rules to enforce on the egress (DiffServ is with respect to the ingress) interface.

The two decisions may very well need to be coordinated to ensure proper provisioning and allocation of network resources. For example, the decision of whether to admit an RSVP flow, or not, would depend on the provisioning policy in place at the egress interface where the flow is leaving the stub network, and vice versa. The issue of combining IntServ and DiffServ to provide an end-to-end QoS solution is discussed in the draft [E2E]. Also, the RSVP WG is currently planning on addressing the use of RSVP within the differentiated services QoS model.

2.2 Interaction between the PDP and the PEP

When a device boots, it opens a COPS connection to its Primary PDP. When the connection is established, the PEP sends information about itself to the PDP in the form of a configuration request. This information includes client specific information (e.g., hardware type, software release, configuration information). During this phase the client may also specify the maximum COPS-PR message size supported (see Section 3.3).

In response, the PDP downloads all provisioned policies which are currently relevant to that device. On receiving the provisioned policies, the device maps them into its local QoS mechanisms, and installs them. If conditions change at the PDP such that the PDP detects that changes are required in the provisioned policies currently in effect, then the PDP sends the changes (installs and/or deletes) in policy to the PEP, and the PEP updates its local QoS mechanisms appropriately.

If, subsequently, the configuration of the device changes (board removed, board added, new software installed, etc.) in ways not covered by policies already known to the PEP, then the PEP sends this unsolicited new information to the PDP. On receiving this new information, the PDP sends to the PEP any additional provisioned policies now needed by the PEP.

3. The definition of the Policy Tree

This section defines data format for the Provisioning client specific information carried in the Decision, Request ClientSI, and Report ClientSI objects. Provisioning client specific data may be defined for the other objects in the future. COPS-PR data is represented by a policy tree containing Policy Rule Classes (PRCs) and Instances of those classes (PRIs), as shown in Figure 5.

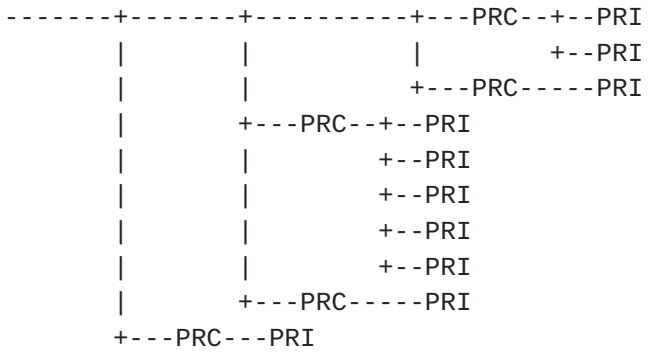


Figure 5: Example of a Policy Tree

The policy tree is based on SMI and MIBs. COPS for RSVP does not need a policy tree, since the information exchanged has a simple format and is defined by existing RSVP objects. COPS for Policy Provisioning needs much more structure, since it needs to represent policies, mappings, Access Control Lists, interfaces, etc.

PRIs (Policy Rule Instances) and PRCs (Policy Rule Classes) have names called PRIDs (Policy Rule IDentifiers). PRIDs have a hierarchical structure of the form 1.3.4.2.7, where the first part identifies the PRC (e.g., 1.3.4) and the last part identifies the instance (e.g. 2.7).

The policy tree names all the policy rule classes and instances and this creates a common view of the policy organization between the client (PEP) and the server (PDP). Therefore, when the PEP receives data from the PDP, the data itself specifies what a PEP is supposed to do with the data. The current granularity of access, i.e., the atomicity of replacement, is proposed as a vector of values.

Note that the PRCs/PRIs in the above diagram are each a vector of values. This proposal is that the hierarchy of PRCs/PRIs is for

benefit of human understanding, not for programmatic understanding, or inheritance.

3.1 Description of the Policy Tree

The Policy Tree is described using SMI and PIBs. SMI and PIBs are defined based on the ASN.1 data definition language [ASN1]. To simplify the implementation and re-use the SNMP encoding/decoding code, the representation of the policy information on the wire must follow BER both for the PRID and for the BPD (BER encoded Policy Instance Data [BER]).

3.2 Operations Supported On a PRI

The following operations are supported on a PRI:

- Install - creates a new instance of a PRC, i.e. a new PRI, or modifies an existing instance. The instance is automatically enabled. Parameters to this operation are a PRID (see Section 4.1) and an "BPD (BER encoded Policy instance Data)" containing the value to assign to the new PRI see (Section 4.2). The BPD specifies all the attributes of the new PRI.

- Delete - This operation is used to delete an instance of a PRC. The parameter is a PRID (see Section 4.1).

3.3 PIB general information

The PIB has a branch that contains general information. Examples of information stored in this branch are:

- TTL (Time To Live): a period of time in seconds. In the event the PEP looses the COPS-PR connection with the PDP, it tries to re-establish the connection with the primary and secondary PDPs. If this fails for a period of time greater than the TTL, the provisioning policies are discarded. The TTL specified in this branch is the default TTL and may be overridden by TTLs present in specific branches. A TTL = 0 means infinite.

- MCMS (Maximum COPS-PR Message Size): an optional maximum message size in bytes. If the COPS-PR client has a fixed MCMS, it must specify it. A value of zero means unknown MCMS.

- Interface to be provisioned.

- Capability information: this may include what filters the PEP supports, what kind of profiles or dispositions it can perform.

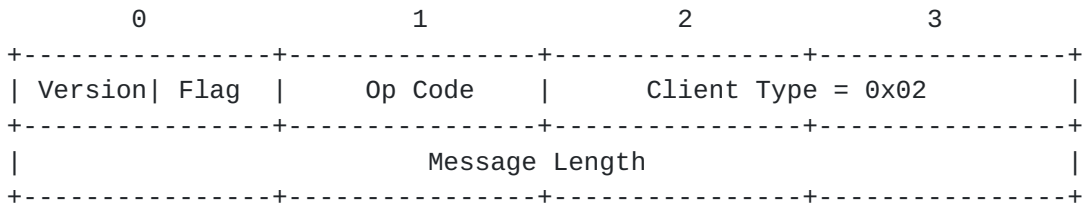
For a more detailed description of the PIB, see [PIB].

4. COPS Policy Provisioning Client Data

The COPS-PR extensions define a new client type:

Client Type = 2; Policy Provisioning Client

Policy Provisioning specific information is sent in a COPS message containing a Common Header with the Policy Provisioning Client type specified:



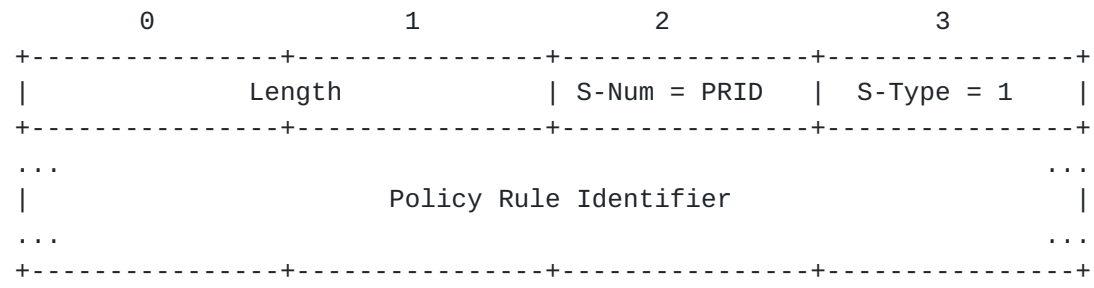
Setting flags to 0x01 implies it is a solicited message.

The COPS protocol specification defines several objects which may carry client specific information between PDP and PEP:

- Context Object (Context)
- Reason code Object (Reason code)
- Decision Object (Decision)
- Error Object (Error)
- Client Specific Info Object (ClientSI) which includes:
  - Request ClientSI
  - Report ClientSI
  - Client-Open ClientSI

4.1 Policy Identifier (PRID)

This object is used to carry the PRID of the Policy Rule Instance to be installed or deleted.

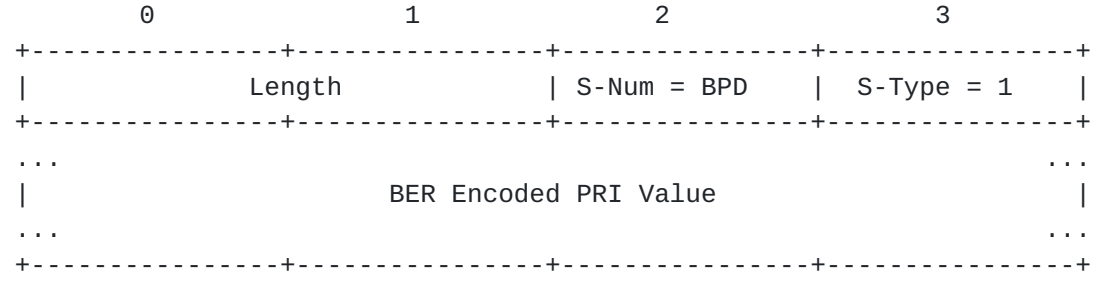


The length is a two-octet value that describes the number of octets (including the header) that compose the object. If the length in octets does not fall on a 32-bit word boundary, padding must be added to the end of the object so that it is aligned to the next 32-bit boundary before the object can be sent on the wire. On the receiving side, a subsequent object boundary can be found by simply rounding up the previous stated object length to the next 32-bit boundary.)

S-Num and S-Type have the same meaning of C-Num and C-Type, but they are ClientSI specific. The value for PRID is S-Num = 1.

4.2 BER encoded Policy instance Data (BPD)

This object is used to carry the value of a Policy Data Instance to be installed, It contains an BER coding of the Policy Data Instance [BER].

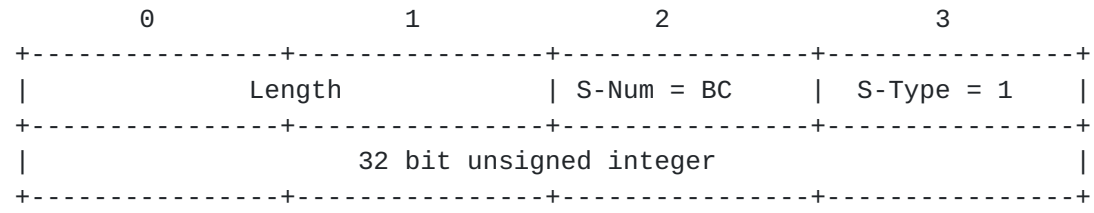


The length is a two-octet value that describes the number of octets (including the header) that compose the object. If the length in octets does not fall on a 32-bit word boundary, padding must be added to the end of the object so that it is aligned to the next 32-bit boundary before the object can be sent on the wire. On the receiving side, a subsequent object boundary can be found by simply rounding up the previous stated object length to the next 32-bit boundary.)

The value for BPD is S-Num = 2.

4.3 Binding Count (BC)

This object is used to specify the number of Bindings that follow.



The length is equal to 8.

The value for BC is S-Num = 3.

4.4 Error Object

The Error object as the same format as in COPS [COPS], but C-Num and C-Type are replaced by S-Num and S-Type. The value of S-Num = 4 and S-Type = 1.

4.5 Policy Provisioning Decision Data

The Policy Provisioning Named Decision Data (<Decision: Named Data>, see Section 5.2) is composed of one or more bindings. Each binding associates a PRID object and an BPD object. The PRID object is always present, the BPD object MUST be present in the case of an install decision and MUST NOT be present in the case of a delete decision.

The BPD object contains the value to be assigned to the PRI that is created or updated.

The Policy Provisioning specific decision data uses the following format:

C-Num = 7
C-Type = 5

```
<Decision: Named Data> ::= <Install Decision> |
                           <Remove Decision>
```

This depends from the <Decision: Flag>, see Section 5.2.

```
<Install Decision>    :: = [<Binding>]+
```

<Binding> ::= <PRID> <BPD>

<Remove Decision> ::= [<PRID>]+

Please note that the delete has the capability of deleting an entire table with a single operation.

4.6 Policy Provisioning Request Data

The Policy Provisioning Configuration request will utilize the COPS Named ClientSI (C-Num=10 C-Type=2) object to carry the same bindings as described above. The Policy Provisioning request Named ClientSI data has the following format:

<ClientSI: Named> ::= < Policy Provisioning Request Data>

<Policy Provisioning Request Data (Named ClientSI)> ::= [Binding]+

4.7 Policy Provisioning Report Data

The Policy Provisioning specific report data is used in the RPT message. The format of the report data is independent on the value of the accompanying COPS Report Type object. Report types can be "Commit" or "No-Commit" indicating to the PDP that a particular set of policies has been either successfully or unsuccessfully installed/deleted on the PEP.

<ClientSI: Named> ::= <Policy Provisioning Report Data>

<Policy Provisioning report data> ::= [<global-error>] [report]+

<global-error> ::= <Error>

where <global-error> is a global error or warning (i.e., not tied to a specific PRID).

<report> ::= <PRID> <specific-error> [<Binding-Count> [<Binding]+]

<specific-error> ::= <Error>

The RPT message can be solicited (in answer to a Decision Data) or unsolicited (e.g., due to a change in the interfaces to be provisioned).

The COPS-PR adds also the following error code:
- 14 Client-Type Specific Error Code;

4.7.1 Commit Data

When used with the "Commit" report type, the objects in the Policy Provisioning Named ClientSI object in the Report Message have the following meaning:

- o <global-error> is a global warning, i.e. an indication of a general warning at the PEP level (e.g., memory low);
- o <specific-error> is a warning indication, i.e. an indication of a warning related to a specific policy that has been installed, but that is not fully implemented (e.g., its parameters have been approximated);
- o <PRID> is a PRID successfully installed/deleted.

4.7.2 No-Commit Data

When used with the "No-Commit" report type, the objects in the Policy Provisioning Named ClientSI object in the Report Message have the following meaning:

- o <global-error> is a global error, i.e. an indication of a general error at the PEP level (e.g., memory exhausted);
- o <PRID> is the PRID of the unsuccessful install/delete;
- o <specific-error> is an error code specific to a binding and is followed by an optional set of <Binding> that caused the error due to conflicts.

In the case of "no commit" the PEP MUST report at least the first error and should report as many errors as possible.

4.7.3 Accounting Data

TBD

5. Message Content

This section describes the COPS messages exchanged between a PEP and PDP for use with Policy Provisioning policy services.

5.1 Request (REQ) PEP -> PDP

The REQ message is used by COPS Policy Provisioning clients for issuing a config request to the PDP, as described in the COPS protocol. The Client Handle is associated with request state originated by the PEP and the PEP is responsible for notifying the PDP when the Handle is no longer in use and can be deleted.

The Policy Provisioning request data, defined above, may be included in the config request from the PEP to the PDP. Currently, the request data is defined for carrying configuration/feature negotiation information from the PEP. This provides the server with information on the types of policy that the interface can enforce and the types of policy data the PEP can install.

The config request message serves as a request from the PEP to the PDP for any Policy Provisioning configuration data which the PDP may have pre-defined for the PEP device, such as access control lists, etc., and any future access data or updates. The pre-configured and any asynchronous Policy Provisioning configuration data can then be sent to the PEP over time via decisions, as decided by the PDP. The configuration information supplied by the PDP is of the consistent client specific format defined above. The PDP responds to the config request with a DEC message containing any available configuration information.

```
   <Request> ::= <Common Header>
                 <Client Handle>
                 <Context = config request>
                 <ClientSI: Named>
```

For <ClientSI: Named> see Section 4.6.

5.2 Decision (DEC) PDP -> PEP

The DEC message (<Decision Message>) is sent from the PDP to a Policy Provisioning client in response to a config REQ received from the

PEP. The Client Handle must be the same Handle that was received in the REQ message. The Client Specific Decision Data for Policy Provisioning clients (<Decision: ClientSI Data>), to be used in the DEC message, is defined in Section 4.5.

The DEC message is sent as an immediate response to a config request with the solicited decision flag set, used to carry pre-defined configuration information set in the PDP, to the PEP. Subsequent DEC messages may also be sent at any time after the original DEC message to continue supplying the PEP with additional/updated policy information. The state carried in the DEC message is correlated with an initial request state by the Client Handle and provides the appropriate PRID information.

Each DEC message may contain multiple decisions. This allows with a single message to install some policies and delete some others. In general a COPS-PR decision message should contain at most one or more deletes followed by one or more install decisions. This is used to solve a precedence issue, not a timing issue: the delete decision deletes what it specifies, except those items that are installed in the same message.

A COPS-PR decision message is also a "transaction", i.e. all the bindings in a message either succeed or fail. This allow to delete some policies only if other policies can be installed in their place. Associating a transaction semantic to a COPS message, instead of defining a specific transaction construct, is not a limitation: in fact the COPS message may have an arbitrary size, only limited either by the PEP memory or by the MCMS parameter, if specified (see Section 3.3).

For each decision (<Decision>), the PEP performs the operation specified in the Decision Flags object (<Decision: Flags>) on the decision data (<Decision: ClientSI Data>]).

```
<Decision Message> ::= <Common Header>
                       <Client Handle>
                       [<Decision>]+ | <Error>

<Decision> ::= <Context>
               <Decision: Command-Code>
              [<Decision: Named Data>]
```

If no configuration state is available when the config REQ is processed by the PDP, a DEC is sent with the "No Configuration Data" decision flag set.

In response to a DEC message, the Policy Provisioning client sends a solicited RPT back to the PDP to inform the PDP of the actual action taken. For example, in response to an Install Decision (see Section 4.5), the PEP informs the PDP if the decision data can be installed, based on the other policy data on the device (are there conflicts, etc.).

5.3 Report State (RPT) PEP -> PDP

The RPT message is sent from the Policy Provisioning client to the PDP to report accounting information from PEP to PDP or notify changes in the PEP (unsolicited report). It is also used as a mechanism to inform the PDP about the action taken at the PEP, in response to a DEC message (solicited report). The Policy Provisioning report data format, as defined above, depends on the Report Type included in the RPT message.

```
<Report State> ::= <Common Header>
                   <Client Handle>
                   <Report Type>
                   [<Policy Provisioning report data>]
```

6. Common Operation

This section describes, in general, typical exchanges between a PDP and Policy Provisioning COPS client.

First, a connection is established between the PEP and PDP and the PEP sends a Client-Open message with the Client-Type = 2, Policy Provisioning client. If the PDP supports the Policy Provisioning client, the PDP responds with a Client-Accept (CAT) message. If the client type is not supported, a Client-Close (CC) message is returned by the PDP to the PEP, possibly identifying an alternate server that is known to support the policy for the Policy Provisioning client.

Once the CAT message is received, the client can send requests to the server. The request a COPS-PR client sends to the server is for configuration information, that is a REQ with "Configuration Request" set in the context object that identifies a specific interface/module and any relevant client specific information (see also Section 3.3). The config request message serves two purposes in COPS-PR. First, it is a request from the PEP to the PDP for any Policy Provisioning configuration data which the PDP may have pre-defined for the PEP device, such as acces control lists, etc. Also, the config request is a request to the PDP to send asynchronous Policy Provisioning

configuration data to the PEP, as it is received by the PDP. This asynchronous data may be new policy data or an update to policy data sent previously.

If the PDP has Policy Provisioning policy configuration information for the client, that information is returned to the client in a DEC message containing the Policy Provisioning client policy data within the COPS Decision object. If no filters are defined, the DEC message will simply specify that there are no filters using the "NULL Decision" Decision Flags object. The PEP MUST specify a client handle in the request message. The PDP MUST process the client handle and copy it in the decision message. This is to prevent the PEP from timing out the REQ and deleting the Client Handle.

The PDP can then add new policy data or update existing state by sending subsequent DEC message(s) to the PEP, with the same Client Handle. The PEP is responsible for removing the Client handle when it is no longer needed, for example when the interface goes down, and informing the PDP that the handle is to be deleted.

For Policy Provisioning purposes, access state, and access requests to the policy server can be initiated by other sources besides the PEP. Examples of other sources include attached users requesting network services via a web interface into a central management application, or H.323 servers requesting resources on behalf of a user for a video conferencing application. When such a request is accepted, the edge device affected by the decision (the point where the flow is to enter the network) must be informed of the decision. Since the PEP in the edge device did not initiate the request, the specifics of the request, e.g. flowspec, packet filter, and PHB to apply, must be communicated to the PEP by the PDP. This information is sent to the PEP using the Decision message containing Policy Provisioning client specific data objects in the COPS Decision object as specified. Any updates to the state information, for example in the case of a policy change or call tear down, is communicated to the PEP by subsequent DEC messages containing the same Client Handle and the updated Policy Provisioning request state. Updates can specify that policy data is to be deleted or installed.

The PEP acknowledges the DEC message and action taken by sending a RPT message with a "Commit" or "No-Commit" Report-Type object. This serves as an indication to the PDP that the requestor (e.g. H.323 server) can be notified that the request has been accepted by the network. If the PEP needs to reject the DEC operation for any reason, a RPT message is sent with a Report-Type of value "No-Commit" and optionally a Client Specific Information object specifying the policy

data that was rejected. The PDP can then respond to the requestor accordingly.

The PEP can report to the PDP the local status of any installed request state when appropriate. This information is sent in a Report-State (RPT) message with the "Accounting" flag set. The state being reported on is referenced by the Client Handle associated with the request state and the client specific data identifier.
Finally, Client-Close (CC) messages are used to cancel the corresponding Client-Open message. The CC message informs the other side that the client type specified is no longer supported.

7. Fault Tolerance

When communication is lost between PEP and PDP, the PEP attempts to re-establish the TCP connection with the PDP it was last connected to. If that server cannot be reached, then the PEP attempts to connect to a secondary PDP, assumed at this time to be manually configured at the PEP.

When a connection is finally re-established, either with the primary PDP or a secondary PDP, the PEP should provide the last PDP address of the PDP for which it is still caching decisions. Based on this information, the PDP may request the PEP to re-synch its current state information (SSQ message). If no decisions are being cached on the PEP (due to reboot or TTL timeout of state) the PEP must not included the last PDP address information. If after re-connecting, the PDP does not request the synchronization, the client can assume the server recognizes it and the current state at the PEP is correct. Any changes state changes which occurred at the PEP while connection was lost must be reported to the PDP in a RPT message. If re-synchronization is requested, the PEP should reissue its configuration requests and the PDP should delete the appropriate PRCs on the PEP (thus, removing all previous decisions below the PRC, effectively resetting all state, and reverting to some static or preconfigured decisions).

While the PEP is disconnected from the PDP, the request state at the PEP is to be used for policy decisions. If the PEP cannot re-connect in some pre-specified period of time (TTL: Time To Live, see Section 3.3), the request state is to be deleted and the associated Handles removed. The same holds true for the PDP; upon detecting a failed TCP connection, the time-out timer is started for the request state associated with the PEP and the state is removed after the specified period without a connection.

8. Security

The use of COPS for Policy Provisioning introduce no new security issues over the base COPS protocol. The use of IPSEC between PDP and PEP, as described in [COPS] is sufficient.

9. References

[COPS] Boyle, J., Cohen, R., Durham, D., Herzog, S., Raja, R., Sastry, A., "The COPS (Common Open Policy Service) Protocol", IETF <draft-ietf-rap-cops-05.txt>, December 1998.

[RAP] Yavatkar, R., et al., "A Framework for Policy Based Admission Control",IETF <draft-ietf-rap-framework-01.txt>, November, 1998.

[E2E] Bernet, Y., Yavatka,r R., Ford, P., Baker, F., Nichols, K., Speer, M., "A Framework for End-to-End QoS Combining RSVP/Intserv and Differentiated Services", IETF <draft-ietf-DiffServ-rsvp-01.txt>, November 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.

[ASN1] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.

[BER] Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), International Organization for Standardization. International Standard 8825, (December, 1987).

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

[PIB] M. Fine, K. McCloghrie, S. Hahn, K. Chan, A. Smith, "An Initial Quality of Service Policy Information Base for

COPS-PR Clients and Servers", draft-mfine-cops-pib-00.txt, February 1999.

10. Author Information

Francis Reichmeyer
Nortel Networks, Inc.
3 Federal Street
Billerica, MA 01821
Phone: (978) 916-3352
Email: freichmeyer@nortelnetworks.com

Kwok Ho Chan
Nortel Networks, Inc.
600 Technology Park Drive
Billerica, MA 01821
Phone: (978) 916-8175
Email: khchan@nortelnetworks.com

David Durham
Intel
2111 NE 25th Avenue
Hillsboro, OR 97124
Phone: (503) 264-6232
Email: david.durham@intel.com

Raj Yavatkar
Intel
2111 NE 25th Avenue
Hillsboro OR 97124
Phone: (503) 264-9077
Email: yavatkar@ibeam.intel.com

Silvano Gai
Cisco Systems, Inc.
170 Tasman Dr.
San Jose, CA 95134-1706
Phone: (408) 527-2690
email: sgai@cisco.com

Keith McCloghrie
Cisco Systems, Inc.
170 Tasman Dr.
San Jose, CA 95134-1706
Phone: (408) 526-5260
email: kzm@cisco.com

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

Andrew Smith
Extreme Networks
10460 Bandley Drive
Cupertino, CA 95014
Phone: (408) 342-0999
Email: andrew@extremenetworks.com

11. Full Copyright Statement