Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."


Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1998). All Rights Reserved.
Abstract

There is a need to represent, manage, and share policies and policy information in a vendor-independent, interoperable, scalable manner. The goal of this document is to provide a framework that meets these needs.

This framework has been developed specifically for QoS traffic management. A secondary goal of this document is to show that this general framework can be extended to other application domains, such as security.

Table of Contents

1. Introduction .................................................... 4
2. Policy Service Functional Blocks ............................... 5
  2.1 The Policy Service ........................................... 6
  2.1.1 The Policy Entry Console .................................. 8
  2.1.2 Policy API ................................................ 9
  2.1.3 The Central Policy Controller ............................. 10
  2.1.4 The Global Conflict Detection Component ............... 11
  2.1.5 The Policy Decision Point ............................... 12
  2.1.6 Conflict Resolution ....................................... 14
  2.1.7 The Message Passing Service Component ................. 14
  2.2 The Directory ............................................... 14
  2.3 The Network Element ......................................... 15
  2.3.1 The Policy Enforcement Point ............................ 16
  2.3.2 Resource Control ......................................... 16
  2.3.3 The Policy Proxy ......................................... 16
3. Roles .................................................................... 17
  3.1 Interface Roles ................................................ 18
  3.2 Interface-Specific Policies ................................... 18
  3.3 Structure of Roles ............................................ 19
4. A Day in the Life of a Policy ................................... 19
  4.1 From the Network Manager to the Policy Service ........... 19
  4.2 From the Policy Service to the Network Element .......... 21
  4.3 Element Name Space .......................................... 22
5. QoS policies .................................................... 23
  5.1 Policy Categories ............................................ 23
  5.2 High-Level Policy Examples .................................. 23
  5.3 A Policy Example ............................................. 24
6. Administration Requirements and Assumptions ................ 26
7. Security Considerations ......................................... 27
Terminology

The terminology specified in this section is in addition to the one presented in the Internet Draft, "Terminology for describing network policy and services" [TERMS].

- Activation condition: the predicate upon which a policy is enabled.
- Classifier: an entity that selects packets based on the content of packet headers according to defined rules.
- CLI: Command Line Interpreter.
- COPS (Common Open Policy Service): a client/server protocol for supporting policy control [COPS];
- Feasibility Conflicts: those conflicts between a new policy and existing policies with regard to the degree of services the network is capable of providing. Feasibility conflicts are mostly determined by the measurement of the state of the network after policy deployment.
- GCD: Global Conflict Detection. The process of examining a policy in the context of all other policies of the Policy Service to determine if there are conflicts between this policy and any other policy already defined. This process does not take into account any actual network state but only the assertions of other policies. Furthermore, it does not target any single object (compare to LCD).
- LCD: Local Conflict Detection. The process of examining a policy in the context of all other policies of the Policy Service to determine if there are conflicts on a specific object between this policy and any other policy already defined. Here, an "object" could be a network device, an interface of a network device, or even an MFC. Compare to GCD.
- MFC: a multi-field (MF) classifier which selects packets based on the content of some arbitrary number of header fields; typically some combination of source address, destination address, DS field
This document presents a framework for the understanding and deployment of policy automation in the network.

Historically, network management has focused on "individual device management" (i.e., setting device parameters on a separate basis). A major drawback of individual device is the lack of a good mechanism to control and coordinate common configuration parameters and commands. That is, given a network-wide policy, how can it be applied in a consistent fashion across multiple network devices?
To overcome this limitation, this framework offers a more scalable alternative to individual device management. This framework provides policy definition, modification, distribution, verification, and administration for a heterogeneous set of network devices (Policy Enforcement Points, or PEPs).

The architecture described in this document may be implemented using standard services and protocols. This document identifies, but does not fully specify, certain additional building blocks, services, and protocols, that are necessary to achieve a scalable architecture.

This framework is mostly concerned with operation in a single administrative domain, i.e., inside a Policy Domain. At the intra-domain level, a single administrative organization creates, stores and enforces a set of policies through a Policy Service. The building blocks of the Policy Service must be constructed so as to share a common view of the policies.

An area for future work is extending this framework to include the exchange and management of policy data between distinct Policy Domains. In general, at the inter-domain level, different Policy Domains do not share a common view of policies. Rather, they cooperate and exchange traffic on the basis of Service Level Agreements (SLAs) using, for example, bandwidth brokers.

To address scalability, this framework advocates logically centralized management and distributed execution. A single Policy Service manages a Policy Domain. However, the Policy Service can be controlled from several consoles. Furthermore, multiple Policy Decision Points (PDPs) can be deployed within the Policy Service, dividing the responsibility for controlling the network elements in the Policy Domain. That is, each PDP can control one or more network elements). This distributed approach accommodates the network’s growing size and growing policy requirements. Furthermore, new network elements that have different functional capabilities can be introduced into a network that uses this framework without having to re-engineer the previously specified policies.

This document concentrates on the application of policies to network devices such as routers and switches, but recognizes that hosts, access servers, firewalls, and other types of devices can also be controlled using the same policies.

2. Policy Service Functional Blocks

Figure 1 illustrates a QoS Policy Domains and its components.
The upper part of the figure (above the dashed line) shows the two main components necessary to implement a Policy Domain, the Policy Service and a Directory. These functions may be hosted on one or more servers and they can be replicated and distributed to increase reliability, response time and availability. The Policy Service and its components are discussed in section 2.1, and the directory is described in section 2.2.

The bottom part of the figure (below the dashed line) shows the network elements and their interconnection. These are discussed in Section 2.3.

2.1 The Policy Service

The Policy Service consists of all of the components necessary for the creation, modification, deletion, management, and distribution of policies to devices within a Policy Domain. Its internal organization is shown in Figure 2.
Components acquire policy information in one of two ways: the complete set of policies in bulk, or incrementally, as policies are changed. The directory is used for bulk transfers. A message passing service is used for incremental updates of policy information and rules, as well as signaling to coordinate the creation, modification, deletion, validation, and installation of policies.
PDPs use policies to configure or answer queries from policy-capable network elements. An example of a query is an RSVP admission request.

The PDP configures policy-capable network elements by using a protocol such as COPS. This may include using policies to control specific mechanisms, such as traffic shaping and queuing. A Policy Proxy may also be used to map COPS commands into specific configuration commands for devices that are not policy-capable (e.g., via CLI or SNMP). This enables the policy service to manage those elements that are not policy capable, but are otherwise capable of providing services that should be managed by the Policy Service.

The presence of multiple PDPs is allowed in this architecture, since it is foreseen that different vendors will propose and/or require their own PDPs. The purpose of this document is to specify a means to achieve better integration and support for more device-specific features from the devices of multiple vendors. Therefore, in a Policy Domain, it is possible to have multiple PDPs from different vendors.

Multiple PDPs may be required even in a Policy Domain whose network devices are all provided by the same vendor. There are two examples of this. First, a PDP may only manage a single aspect of policy (e.g., just network QoS, as opposed to security). In this case, some entity other than the PDPs must coordinate policies delivered by the multiple different PDPs to a single device. [See Section 2.1.3, the Central Policy Controller, below]. Second, there may be a PDP in each of several sites of an enterprise or a Service Provider. Each such PDP has local knowledge. It uses this local knowledge to validate policies as well as to manage its local elements. This local knowledge could be due to physical, geographical, or technical constraints (e.g., different types of devices). Therefore, since each PDP only controls a portion of the devices in the network (e.g., those network elements that it has knowledge of), multiple PDPs are required.

2.1.1 The Policy Entry Console

The Policy Entry Console (PEC) is the component used by the network administrator to input, modify, store, view, search, and delete policies and related support information. The PEC reads policy information from the Directory in order to create an overall view of the policies and policy information (according to the permissions of the PEC operator). The PEC then interacts with the operator, permitting him/her to view and modify various aspects of policy.

When a policy is created, modified, or deleted, the PEC sends the
change to the Central Policy Controller (CPC). The CPC evaluates the policy changes to ensure that the results of the policy operation are valid. For example, the CPC can check to ensure that deleted policies do not create an inconsistent state, and that new and modified policies are correctly constructed (and that they don’t conflict with other policies).

The PEC SHOULD present a notification to the operator, indicating that the policy change was either accepted or not accepted. If the policy change was not accepted, then the PEC may present additional error information that the CPC found to the operator for correction. The kinds of errors include:

- conflicts that due to overlapping conditions and actions that the administrator must resolve;
- other errors generated by other components of the Policy Service that the administrator must repair.

If the policy is accepted by the CPC, the administrator may direct the CPC to make it persistent. Making a policy persistent includes storing it in the directory and informing the PDPs of the fact that it was made persistent.

As part of its normal operation, the PEC must maintain an up-to-date view of the policies in which its users are interested.

2.1.2 Policy API

The Policy API (PAPI) is part of the exported interface of the Policy Service that enables programs to participate directly in the creation, editing, installation, removal, and management of policy. It provides an interface through which programs interact with the Policy Service. Services provided by the PAPI include the following:

- Authentication/Authorization -- authenticate the user on whose behalf the program using the PEC is operating, and determine what operations this user is allowed to perform.
- Policy System Configuration -- provide an interface to show how components of the policy system are configured, and enable those configurations to be changed.
- Management of the message passing interface -- includes registering with a message passing service either as a publisher, subscriber, or both, as well as sending and receiving messages, and other
necessary but less important functions.

- Construction of the messages to be sent from a PEC or a program via the messaging system.

The messages themselves are used to perform various operations, including the following:

- Change (Add/Modify/Delete) policy;
- Enable/disable policy;
- Return codes indicating the success or failure, and optionally the reason, of attempted policy change messages. These return codes include:
  - change accepted (by which PDPs);
  - change rejected (by which PDPs) and the reason(s) why;
  - change accepted conditionally with the following reservations;
  - change has the following list of conflicts with these policies.

Note that PAPI clients have read-only access to the directory describing policies and policy information. The only component of the Policy Service that can write to the schema is the CPC.

In addition to the PEC, applications and programs written to perform, for example, automatic policy addition, modification, or deletion, as well as the exchange of policy information, also use the PAPI. It is hypothesized that one of the programs written to use the PAPI is a scripting language. This provides a means for network managers to operate on policies in a more programmatic way.

2.1.3 The Central Policy Controller

The Central Policy Controller (CPC) is the single logical point through which all policy changes are passed. It is may be comprised of several physical entities in a Policy Service. However, in this case, they must all cooperate so that the effect of policy change in the Policy Domain is both predictable and consistent. That is, the CPC ensures that policy change -- even change initiated simultaneously from several sources -- proceeds in an orderly manner. In the physically distributed case, the CPC appears as a single logical entity.
The CPC is required because a predictable policy change process, across multiple independent components, is an integral part of the Policy Service. In particular:

- PDPs must be informed of proposed policy changes so that they can validate them and perform satisfiability and feasibility checking (refer to section 2.1.5).

- PDPs must have a place to which they can respond with their evaluation of the proposed policy change.

- PAPI clients require a consolidated response to their submitted policy change requests.

Additional detail describing how the CPC functions in the Policy Service is given in section 3.

2.1.4 The Global Conflict Detection Component

The Global Conflict Detection (GCD) component of the CPC is used when, via the PAPI, a policy change is submitted to the CPC. Part of the general verification process of ensuring that the input policy change request is acceptable is determining if the new policy globally conflicts with any other policy in the Policy Domain.

Global conflicts are those based on the properties of the policy and not the specific devices (or their interfaces) to which the policy might apply (see section 2.1.5, Local Conflict Detection). Two policies globally conflict when all their conditions are satisfied, but one or more of the actions of one policy conflict with one or more of the actions of another policy. The conditions of two policies are both satisfied when their criteria are both met simultaneously. The actions of two policies conflict when they cause different operations to be applied to the same resource. Referring to [SCHEMA], this can happen only if their corresponding conditions are all satisfied. For example, if Policy A specifies that traffic should be forwarded for a particular source IP address, but Policy B specifies that traffic should be denied for that same source IP address, these policies will conflict if each of their conditions are all satisfied. For example, if packets from Joe are supposed to receive a DSCP of 6, and FTP packets are supposed to get a DSCP of 8, what happens when Joe sends FTP packets?

Global Conflict Detection does not rely upon the attributes and/or the state of any particular network device(s) for conflicts to exist.
Often, policies that appear to conflict globally actually do not conflict with each other. This is because the administrator may intend to use these policies at different times, such as in response to different network conditions. The policies do not conflict because there are additional factors that are not specified as part of the conditions that ensure that the policy conditions are non-intersecting. An example of such a usage is given in section 5.

2.1.5 The Policy Decision Point

The Policy Decision Point (PDP) is the Policy Service component that actually communicates with policy-capable network elements. For network devices that are not policy-capable, a Policy Proxy is used to translate between their communication and control capabilities and those of the PDP. This is where device configuration changes and policy decisions are made to implement abstract policies.

The following two sections describe how device configuration changes and policy decisions are made in the PDP.

2.1.5.1 Device Configuration Changes

Modification of policies may end up changing the configuration of some network elements. The PDP determines which devices are effected by a particular modified policy. The PDP is also responsible for understanding the policies currently installed in the devices that it controls and mapping policies (both new as well as changes to existing policies) into specific configuration and/or state changes of the device. These can both be accomplished through, for example, the COPS protocol, possibly with the Policy Proxy as a mediator, translating the COPS messages into (for example) CLI or SNMP device configuration commands. Some examples for such policies and their outcomes are:

- Creating a new policy may end up changing the queuing mechanism of some interfaces from FIFO to WFQ.
- A policy to grant high priority to one application may end up installing a MFC on the edge switch near that application server to color financial month-end traffic based on the source IP address and/or port number.
- Limiting Pointcast to 20% of all WAN traffic may install a MFC on some interfaces to appropriately shape Pointcast traffic.
2.1.5.2 Real-time Response to Policy Queries

Network elements may consult with their PDP in order to decide which QoS requests to admit and what special treatment to apply to specific flows. The queries can be initiated in several ways, including:

- An RSVP request is received by a network element. The network element queries the PDP regarding the request. When a request is accepted, the PDP can also change various parameters to be included in the message forwarded to the next hop. This may include some policy elements and optionally the requested flow specification. The decision may be based on all available RSVP parameters as well as applicable aggregate limitations.

- A new flow may match some predefined MFC that directs the device to refer the flow to the PDP. The PDP can decide to assign a DS value \([\text{DSFIELD}]\) to this particular flow, and more generally, to push new MFCs to the device (e.g. so future flows with the same characteristics will get the same treatment).

The PDP may be partially or completely implemented on a network device in order to make the policy decision process more efficient and to increase scalability. In this case, the network device is responsible for making some or all decisions on its own.

2.1.5.3 Local Conflict Detection Component

The Local Conflict Detection (LCD) component is an integral part of the PDP. Whereas the GCD checks for policy conflicts that do not apply to any specific network device, the LCD checks for policy conflicts that apply to all network devices that are controlled by its PDP or Policy Proxy.

The LCD detects local conflicts and checks for satisfiability and feasibility of a policy (new, changed, or deleted) in which this PDP has interest. The types of checks performed include:

- **Conflict Detection.** This entails checking that the new, modified, or deleted policy does not conflict with any existing local policy.

- **Satisfiability.** This is a set of checks to ensure that the resources needed by a policy, in isolation from all other local policies, are available in the devices to which this policy applies. For example, suppose that a policy requires that a certain
set of paths through the network (via the devices that this PDP manages) provide a certain specific queuing behavior. Suppose further that on one of the paths at one of the interfaces, no advanced queuing mechanisms are available. This would mean that the needs of the policy are not satisfied. Thus, the policy itself is not satisfied, implying that this policy cannot be implemented in these devices.

- Feasibility. This compares the available services of the network with respect to the full set of policies that want to use those services. Feasibility checking will most likely require post-policy deployment checking that is sensitized to the particular network elements involved as well as the nature and effects of the deployed policies. This is beyond the scope of this document.

2.1.6 Conflict Resolution

Conflict resolution is not currently a part of this framework. In general, when a conflict occurs, the source of a proposed policy that triggers the conflict will be notified.

2.1.7 The Message Passing Service Component

The Message Passing Service is a publish-subscribe system that communicates using events. It is used for incremental updates to policies as well as to signal the coordination of the creation, modification, validation, administration, management, and installation of policies.

The following components make use of the Message Passing Service:

- PAPI (Policy API)
- CPC (Central Policy Controller)
- PDP (Policy Decision Point)
- Policy Proxy

2.2 The Directory

A Policy Service is comprised of multiple diverse components that may be geographically distributed. These components must share and edit common data. The Directory is well-suited for serving as a common
unifying repository that enables these components to share, modify and use common data.

The Policy Service manages three fundamentally different types of data: static, low-latency, and transient. While the directory easily handles static data, it does not scale to large systems for low latency data, and cannot handle transient data at all.

Therefore, the Policy Service will employ a Directory to serve as the repository for all common, static data and some low-latency data.

An advantage of using the Directory as the centralized backing store for the Policy Service is that most organizations already store user, computer, file server, and other network resources in the Directory. This enables the Policy Service to specify policies in terms of users and applications that use the network.

The Policy Service uses the Directory to store and locate the following information:

- users and network resources that it wants to apply policies to;
- definitions of policy conditions and actions, as defined in [SCHEMA];
- the canonical definition of a policy rule and a policy group (e.g., a set of policy rules), as defined in [SCHEMA];
- global settings that are used in the configuration of QoS (e.g. thresholds, weights, etc...);

LDAPv3 is the protocol used to access the Directory.

### 2.3 The Network Element

This section describes the architecture of Network Elements that are compliant with this architecture. Formally, a network element is compliant if it can either properly represent itself directly to the PDP as described below, or if it can be represented by a Policy Proxy and provide the services required by the Policy Service (see Section 2.3.3).

Input to the network element comes from either the PDP or the Policy Proxy. The network element receives the policy information and/or policy decision through, for example, its COPS interface, and may optionally store a policy decision and/or information locally. (This
is an optimization to enable the network element to not have to ask either the PDP or the Policy Proxy when the same type of policy decision is later encountered.) The network element then maps the data supplied to its specific mechanisms. This is done either directly or indirectly using another protocol and/or API.

2.3.1 The Policy Enforcement Point

The Policy Enforcement Point (PEP) is the point in the network element that enforces the policies through whatever hardware and software means are appropriate.

Note that network elements support different types of QoS mechanisms. These mechanisms can be available either in software or in hardware. Therefore, policies must be abstract enough to allow for those differences, while remaining specific enough to provide useful direction to the PEP (see Section 3).

2.3.2 Resource Control

Resources that are used to implement policy from a Policy Service must be under complete control of the Policy Service. However, it is a fact of life that simpler control mechanisms, such as telnet/CLI and SNMP, will continue to be used. The problem is that the Policy Service may not be aware of changes made to devices under its control by such mechanisms. Therefore, it is recommended that such mechanisms be disallowed (or at least temporarily blocked) for the duration that the Policy Service is using a resource. Once a policy is terminated (either because the flow is completed or the policy's time to live expires), the Policy Service may relinquish control of the resource and allow other mechanisms to control it. This is necessary so that the Policy Service is the only entity that can control application of policies to a network element. Also note that this document does not recommend a way to either enforce this recommendation, or notify the Policy Service that this recommendation is being violated. Both of these issues are beyond the scope of this document.

2.3.3 The Policy Proxy

Policy Proxies are used in cases where network elements that implement policy decisions are unable to implement the protocols necessary to exchange messages with the PDP (e.g., COPS). The two typical cases where Policy Proxies will be used are legacy devices and network elements that are memory-space constrained,
either by cost or device size. The specifics of how a Policy Proxy communicates with the devices it represents is beyond the scope of this document, but typical examples include SNMP, telnet/CLI, and/or RPC.

A Policy Proxy will only use the DiffServ model of interaction between itself and the Policy Service [COPS-PR], not the RSVP (IntServ) model. The configuration of the proxy is beyond the scope of this draft.

3. Roles

Some policy data is global - it applies everywhere in the network. Other policy data is location-specific - it applies at one and only one specific location. In addition, it is useful to have a level of granularity between these two extremes. The concept of "roles" is used to provide this intermediate level of granularity.

A role is a means of grouping together a set of objects, so that one or more policies can be specified as being applied to the entire group of objects. This idea is not new; for example, it has been used to have multiple users belong to a group, and to attach user profiles, privileges, and permissions to the group, so that each user in the group gets those profiles, privileges, and permissions. It has also been used extensively in database management systems. For policy data, the idea is applied by assigning policies to "roles" and assigning "roles" to network components (devices, interfaces, etc.). "Roles" provide a powerful method of indirection, since:

- new policies are specified for a role, instead of having to specify them for each and every individual network component to which they apply.
- the modification of existing policies is specified within a role, instead of having to modify them for each and every individual network component to which they apply.
- existing policies are applied to a newly-installed network component by assigning the relevant roles to the new component, rather than copying policies from existing components to the new component.
- operators are encouraged to generate network-wide policies, rather than having to remember all the individual components to which they should be applied.
neither the permanently-stored policy data, nor the Policy Service, needs to have intimate knowledge of each and every device (let alone each and every device interface!) in the network; rather, each device can inform the Policy Service of the roles for which it needs policy data.

Policy management and communication traffic is greatly minimized.

Examples of potential roles are: "backbone router", "interface to a (named) policy domain", "high-bandwidth interface", "Frame-Relay interface", etc.

3.1 Interface Roles

When a device connects to the Policy Service, it supplies the roles assigned to its interfaces to the Policy Service. This enables the Policy Service to download only that subset of policies that are relevant for those roles (i.e., omit policies which are not needed by the device).

Some devices provide QoS mechanisms that can be configured to operate differently on different interfaces. In this case, roles enable different QoS policies to be applied to different interfaces, and there is no inter-dependence of what roles are assigned to what interfaces. Other devices have centrally-implemented QoS mechanisms that operate independent of a packet’s ingress/egress interface. In this case, interfaces served by such centrally-implemented QoS mechanisms must take the same set of QoS policies. This can also be ensured by using roles to represent such policies. The use of the same roles to control interfaces that share the centrally implemented QoS mechanisms simplifies their management and reduces the chance for configuration errors.

3.2 Interface-Specific Policies

Roles can also be used as the mechanism to specify location-specific policies on a per-interface basis. In particular, a role that is assigned to at most one interface at a time defines interface-specific policies for the interface to which it is assigned. This use of interface-specific roles allows the permanently stored policy data to be independent of the network configuration. For example, the permanently stored policy data can be applied to the current network topology based on the assignment of roles to interfaces, as opposed to requiring specific knowledge about a device or its interfaces. Equivalently, an interface-specific role can be thought of as
applying policy to a logical interface that is mapped to a physical interface through device configuration.

3.3 Structure of Roles

This document does not specify the semantic meaning or structure of roles. We recognize, however, that in order to reduce the complexity of large numbers of roles (both for the system as a whole and to download to devices), it will be necessary for PDPs to impose some structure on roles. It remains an open question as to whether that structure must be expressed to the PEP. An example of a structure a PDP might impose would be that there are roles of type IPSEC-tunnel, which take as parameters an access-list, a tunnel end point, and a crypto-map.

4. A Day in the Life of a Policy

This section provides two example scenarios of how policies are treated in a Policy Service.

4.1 From the Network Manager to the Policy Service

This section describes a "day in a life" of a QoS policy from the perspective of the network manager installing a new policy, modifying an existing policy, or deleting an existing policy.

1. The Network Manager creates or modifies a policy, using either the PEC, an ad hoc program, or a third party application. The policy is passed in a message to the PAPI. The message contains the type of policy (and other parameters that characterize the policy) and the policy state (new, modified, deleted).

2. The PAPI sends the policy message to the CPC. The CPC establishes a state machine to track the state of the policy being changed.

3. The CPC uses the GCD to perform abstract conflict checking. Abstract conflict checking is the process of determining if the proposed policy conflicts with other policies in a general way (as opposed to a specific device interface). If conflicts exist, the policy is returned to the PEC for resolution.

4. The CPC sends the changed policy on to all PDPs registered for this type of policy (using the Message Passing Service).
5. Each participating PDP performs appropriate checking on the policy. This may include:

- Local conflict checking (e.g., against a specific interface) of this policy against other policies this PDP is handling;
- Satisfiability of the policy by the portion of the network under the management of this PDP;
- Feasibility of this policy in the context of other policies and the portion of the network that this PDP manages.

6. Each PDP sends a response message with the PDP’s evaluation of the policy back to the CPC. Possible replies include:

- Policy is OK
- Policy is not understood
- Policy cannot be implemented because of one or more specific reasons (possible reasons include:)
  - Requires resources that this device does not have
  - Requires resources that are not known
  - This policy conflicts with policy (name and/or ID to be passed back in the response) on actions (to be supplied in the response)
  - Policy is not relevant to this device (e.g., asked for control of a resource that this device does not have)

7. CPC receives the PDP response messages. It evaluates them and generates reply to originator of change (PEC or other process). Possible replies include:

- Policy is acceptable
- Policy is acceptable with the following exceptions (derived from the replies)
- Policy is not acceptable because of the following reasons (derived from the replies)

8. If the CPC reply indicates that the policy is acceptable, then the PEC (or policy change agent) may request that the policy be
installed in one or more of the devices that each PDP governs. If the CPC reply indicates that the policy is not acceptable, then the policy cannot be installed, and the process is complete. (The PEC or application submitting the policy must decide what to do about policies that have been rejected. For example, it might modify the policy and resubmit it.)

9. If the policy is acceptable, then an installation message is sent to the CPC. The CPC then instructs the applicable PDPs to install the policy to the (sub)set of devices that each PDP controls. If a PDP encounters an error, then it informs the CPC, which in turn informs the PEC or the policy change agent, and the process starts over.

### 4.2 From the Policy Service to the Network Element

The following section describes a "day in a life" of a QoS policy from the perspective of the Policy Service installing a new policy, modifying an existing policy, or deleting an existing policy to a Network Element.

1. When a device reboots, it opens a TCP connection to its Primary PDP.

2. When the connection is established, the device sends information about itself to the PDP. This information includes the set of unique roles that its interfaces have.

3. In response, the PDP downloads all provisioned policies that are currently relevant to that device.

4. On receiving the provisioned policies, the device maps them into its local QoS mechanisms, and installs them.

At this point, several different scenarios can happen. The first scenario is that conditions may change in the network that effect one or more devices that are under the control of a PDP (or its Policy Proxy). In this case, the PDP must determine if any corresponding changes are required in the provisioned policies currently in effect in any of the devices that it controls. If changes are required, the PDP sends the changes (installs, modifies, or deletes) in policy to those devices, and each device updates its local mechanisms appropriately.

The second scenario involves RSVP. It is as follows:
1. If the device is running RSVP, then the device sends a request to the PDP (or its Policy Proxy) whenever it receives a new RSVP Path or Resv message, asking the PDP how to respond.

2. On receiving a request for an RSVP-based policy decision, the PDP determines whether the specified RSVP request is permitted, and responds accordingly.

3. On receiving an RSVP-based policy decision, the device processes the waiting RSVP request accordingly, either rejecting or accepting it.

The third scenario involves changes to roles on a device controlled by a PDP (or its Policy Proxy). It is critically important for the device to send a new set of roles to its PDP (or Policy Proxy) if the roles of a particular interface are modified. This is because roles are used to identify what policies are applicable to a given network device. On receiving the set of new roles, the PDP sends any additional policies now needed to the device.

The final scenario involves a change to the configuration of a device under control of a PDP or a Policy Proxy that was NOT a result of a policy change. Examples include adding or removing a board, installing new software, and other changes that effect the device. In this case, the CPC must be informed of such changes. The device could do this directly, or it could ask its PDP or Policy Proxy to inform the CPC for it. The specific means is beyond the scope of this document; however, it is important for the Policy Framework to account for this possibility and define what to do when it occurs.

4.3 Element Name Space

In order for a Policy Service to implement a policy, it must be able to address all elements within its domain. Such names must, again, be abstract enough to allow for representations other than internal (e.g., Serial1/0/0) but specific enough to identify the service being offered (i.e., an SNMP-like index table that contains sufficient identifying information). This document does not specify the knowledge model under which a Policy Service determines where a policy needs to be implemented. However, once it has identified a resource, it needs a way to reference it. In particular, it proposes a simple mechanism - that of using a role to identify, for example, a device interface - as one way of defining such a network element name space.
5. QoS policies

This section describes several attributes of policy:

- Policy Categories;
- Policy Organization;
- High-level Policy Examples.

5.1 Policy Categories

This framework specifies a structure of a policy that may be implemented as specified in [SCHEMA]. That document describes the organization of what this document calls policy. This organization consists of a set of foundational base classes that represent a policy rule, a group of policy rules, a condition, and an action. These classes define a mapping that is suitable for implementation in a directory that uses LDAPv3 as its access protocol. That document also describes different categories of policies (see [SCHEMA], section 2.0 _ Modeling Policies, for more specific information).

5.2 High-Level Policy Examples

In general, the QoS policies entered by the network manager have two important attributes: they are high-level and network-wide. Examples of high level policies are:

- the traffic generated by the Stock Exchange service has very high priority;
- the data traffic generated by the NetMeeting application has low priority;
- the voice and video traffic generated by NetMeeting has high priority and it has real time needs;
- the traffic generated by the company CEO has high priority;
- the traffic toward the SAP server has high priority;
- the traffic generated by Internet-News has very low priority.
5.3 A Policy Example

Suppose that we want to define a set of usage policies that govern the implementation of the goal "Under all circumstances, traffic to and from the Trading Services are to have precedence over all other types of traffic". Let's suppose that there are three general network circumstances under which we wish to define this goal:

- Normal traffic conditions -- all network paths are completely functional;
- Degraded traffic conditions -- some network paths are not functional;
- Catastrophic traffic conditions -- there is a financial crisis to be dealt with.

For each of these cases, we establish coloring policies that govern the treatment of the classes of traffic we wish to permit under the corresponding case. We do not define what determines the condition of the network. We only suppose that the condition can be determined and that some agency can communicate with the Policy Service that determination.

Suppose that we have the following classes of traffic in the target system:

- Trading Service traffic
- E-Mail
- FTP

Further, suppose that the network has been engineered to provide three classes of service through appropriate queue definitions. Finally, suppose that the following actions are to be taken under the various conditions of the network:

<table>
<thead>
<tr>
<th>Network Condition</th>
<th>Trading Traffic</th>
<th>E-Mail Traffic</th>
<th>FTP Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>DSCP == 6</td>
<td>DSCP == 4</td>
<td>DSCP == 2</td>
</tr>
<tr>
<td>Degraded</td>
<td>DSCP == 6</td>
<td>DSCP == 2</td>
<td>DROP</td>
</tr>
<tr>
<td>Catastrophic</td>
<td>DSCP == 6 DROP</td>
<td>DROP</td>
<td>DROP</td>
</tr>
</tbody>
</table>

where DSCP is the DS Code Point [DSFIELD].
Then, the following might be the policy defined to handle the above situation:

The goal for the following sets of policy is to assure the following:

"Under all circumstances, traffic to and from the Trading Services are to get through"

There are three coordinated set of policies needed to satisfy this goal, corresponding to the three different network conditions that are anticipated. Note that the following policies are inherently conflicting if they are applied simultaneously, but in fact they are not conflicting since only one set of policies is applied for one of the three conditions. Treating them as a group preserves the semantics and purpose to which they were designed (e.g., as a response to a specific network condition, like financial crisis).

The first set of policies, to be applied only under normal conditions, is:

Policy 1a:
Source == "Trading" or Destination == "Trading" --> DSCP <- 6;
Policy 2a:
Source == "E-Mail" or Destination == "E-Mail" --> DSCP <- 4;
Policy 3a:
Source == "FTP" or Destination == "FTP" --> DSCP <- 2.

The second set of policies, to be applied only under degraded conditions, is:

Policy 1b:
Source == "Trading" or Destination == "Trading" --> DSCP <- 6;
Policy 2b:
Source == "E-Mail" or Destination == "E-Mail" --> DSCP <- 2;
Policy 3b:
Source == "FTP" or Destination == "FTP" --> drop;

Finally, the third set of policies, to be applied only under catastrophic conditions, is:

Policy 1c:
Source == "Trading" or Destination == "Trading" --> DSCP <- 6;
Policy 2c:
Source == "E-Mail" or Destination == "E-Mail" --> drop;
Policy 3c:
Source == "FTP" or Destination == "FTP" --> drop.

We assume that all ingress and egress ports over which these three types of traffic flow are marked with the role "TS" (Trading Service). We also assume that each of the policies is marked as being part of the role "TS".

Inspection of these policies reveals that all there are many conflicts. In particular each set of policies (numbered 1, 2, or 3) conflicts. However, because the administrator never intends for all of these policies to be enabled at the same time, s/he instructs the system to ignore the conflicts. Then, when network conditions are Normal, the administrator enables the first set of policies and disables the second and third sets. When network conditions are degraded, s/he enables second set and disables the first and third. Finally, if a catastrophic event occurs, s/he enables the third set and disables the first and second.

This example has shown that the ability to specify conflicting conditions (e.g., the sets of policies) helps describe the overall solution as to how the goal of ensuring that Trading Service traffic will always have priority over all other services in the network.

6. Administration Requirements and Assumptions

This framework makes certain assumptions about device configuration. For example, certain tasks are performed either by a network administrator or by some administrative tool not defined in this document (an example of the latter would be a router configuration tool that interactively assists an administrator to construct a usable configuration file). The following tasks are identified as administrative requirements (though they are not specified in this document):

- Entry of policies. Policies are either entered via the PEC or communicated from another system via the PAPI.

- Configuration of roles within elements. It will be necessary for the administrator or some tool not specified by this framework to identify what role a resource (such as an interface) will play in the network. For example, it is not possible for the policy service to know that interface Serial0/0/1 plays the role "ABCcorp".

- Configuration of proxies. How a proxy translates (for example) COPS transactions into non-compliant device-specific communication may require some additional configuration. Specification of policy
domain members. It is necessary for an administrator to decide which policy domain a particular element will reside in (e.g., which PDP or Policy Proxy is responsible for providing policies to the device and making policy decisions on behalf of the device). This can be accomplished by (for example) configuring PDP (or Policy Proxy) addresses in network elements.

7. Security Considerations

Security and denial of service considerations are not explicitly considered in this version of this document. However, the policy architecture must be secure as far as the following aspects are concerned. First, the mechanisms proposed under the framework must minimize theft and denial of service threats. Second, it must be ensured that the entities (such as PEPs and PDPs) involved in policy control can verify each other’s identity and establish necessary trust before communicating. The architecture defined in this document MUST not compromise either of these goals.

8. Intellectual Property

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF’s procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11.

Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.
9. Acknowledgments

We would especially like to thank Keith McCloghrie, Stephen Schleimer, Eliot Lear, and Michel Langlois for their helpful comments.

10. References


11. Author’s Addresses:

Silvano Gai  
Cisco Systems  
170 West Tasman Drive  
San Jose, CA, 95134-1706  
Tel.: (408) 527.2690  
Email: sgai@cisco.com

John Strassner  
Cisco Systems  
170 West Tasman Drive
San Jose, CA, 95134-1706  
Tel.: (408) 527.1069  
Email: johns@cisco.com

David Durham  
Intel  
2111 NE 25th Avenue  
Hillsboro, OR 97124  
Tel.: (503) 264.6232  
Email: David_Durham@mail.intel.com

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

Hugh Mahon  
Hewlett-Packard  
3404 East Harmony road, MS A2  
Fort Collins, Colorado 80528-9599  
Tel.: (970) 898.2487  
Email: h_mahon@fc.hp.com

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

12. Full Copyright Statement

Copyright (C) The Internet Society (1997).  All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other

Gai, Strassner, Durham, Herzog, Mahon, Reichmeyer
Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.