Internet Engineering Task Force
INTERNET DRAFT
23 June 1999

John Strassner
Cisco Systems
Ed Ellesson
IBM

Terminology for describing network policy and services

draft-ietf-policy-terms-00.txt

Status of Memo

Abstract

Recent work in the IETF has led to multiple protocols which support the classification of packets for the purposes of treating certain classes or flows of packets in a particular way compared to other packets. The successful wide-scale deployment of these protocols depends on the ability to administer and distribute consistent policy information to the multiple devices in the network which perform the classification and packet conditioning or treatment. As a result, there is a clear need to develop a scalable framework for policy administration and distribution that will enable interoperability among multiple devices and device types that must work together to achieve a consistent implementation of policy.

Unfortunately, terms like "policy" and "service" are not currently defined in sufficient detail as to enable the definition, specification, and implementation of policy servers and how policy is recognized and enforced at the device level. At present, both "policy" and "service" (as well as other related terms) are overloaded with multiple (often conflicting) meanings. This makes communication about policy in general and specifically policy-based networking cumbersome and difficult. In addition, information modeling approaches, like CIM [CIM], define specific meanings for these terms that must be coordinated with the IETF community.

This document defines a set of terms that the Internet community can use to exchange ideas on how policy creation, administration, management, and distribution could work among policy servers and multiple device types.

Definition of Key Word Usage

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in RFC 2119 [TERMS]. These words will be capitalized to ensure that their intent in this context is easily discernible.

Table of Contents

Table of Contents (continued)

1. Introduction and Motivation

Recent work in the IETF has led to protocols which support the classification of packets for the purposes of treating certain classes or flows of packets in a particular way compared to other packets. The purpose of such classification may include preferential queuing or dropping, admitting or denying access, or encrypting the packet's payload, to cite just a few examples. Protocols that explicitly support some or all of these functions include COPS, RADIUS, RSVP, and IPSEC. In addition, the IntServ [ISWG], DiffServ [DSWG], RAP [RAPWG], ISSLL [ISSLLWG] and IPSEC [IPSECWG] working groups are defining approaches to using these protocols.

The successful wide-scale deployment these and other protocols depends on the ability for the administrator of a network domain to administer and distribute consistent policy information to the multiple devices in the network which perform the classification and packet conditioning or treatment. Protocols that could be used for the distribution of the policy include LDAP, COPS, SNMP, and TELNET/CLI. The multiple types of devices that must work in concert across even a single domain to achieve the desired policy can include hosts (clients and servers), routers, switches, firewalls, bandwidth brokers, subnet bandwidth managers, network access servers, and policy servers, to name just a few.

As a result, there is a clear need to develop a scalable framework for policy administration and distribution that will allow interoperability among the multiple devices and device types that must work together to achieve a consistent implementation of the network administrator's policy. Unfortunately, terms like "policy" and "service" are not currently defined in sufficient detail as to enable the definition, specification, and implementation of policy servers and how policy is recognized and enforced at the device level. At present, both "policy" and "service" (as well as other related terms) are overloaded with multiple (often conflicting) meanings. This makes communication about policy in general and specifically policy-based networking cumbersome and difficult.

This document defines a set of terms that the Internet community can use to exchange ideas on how policy creation, administration, management, and distribution could work among policy servers and multiple device types.

2. Previously Defined Terminology

The following terms have been previously defined in other Internet Drafts and are used in this draft to better define policy and policy-based networking terms.

Definitions taken from RFC2474 (Definition of the DS Field):

Classifier: an entity which selects packets based on the content of packet headers according to defined rules.

Differentiated Services Boundary: the edge of a DS domain, where classifiers and traffic conditioners are likely to be deployed. A differentiated services boundary can be further sub-divided into ingress and egress nodes, where the ingress/egress nodes are the downstream/upstream nodes of a boundary link in a given traffic direction. A differentiated services boundary typically is found at the ingress to the first-hop differentiated services-compliant router (or network node) that a host's packets traverse, or at the egress of the last-hop differentiated services-compliant router or network node that packets traverse before arriving at a host. This is sometimes referred to as the boundary at a leaf router. A differentiated services boundary may be co-located with a host, subject to local policy.

Differentiated Services-Compliant: in compliance with the requirements specified in [RFC2474].

Differentiated Services Domain: a contiguous portion of the Internet over which a consistent set of differentiated services policies are administered in a coordinated fashion. A differentiated services domain can represent different administrative domains or autonomous systems, different trust regions, different network technologies (e.g., cell/frame), hosts and routers, etc.

Differentiated Services Field: the IPv4 header TOS octet or the IPv6 Traffic Class octet when interpreted in conformance with the definition given in this document (RFC2474).

Mechanism: The implementation of one or more per-hop behaviors according to a particular algorithm.

Per-hop Behavior (PHB): a description of the externally observable forwarding treatment applied at a differentiated services-compliant node to a behavior aggregate. The description of a PHB SHOULD be sufficiently detailed to allow the construction of predictable services, as documented in [RFC2475].

Per-hop Behavior Group: a set of one or more PHBs that can only be meaningfully specified and implemented simultaneously, due to a

common constraint applying to all PHBs in the set such as a queue servicing or queue management policy.

Traffic Conditioning: control functions that can be applied to a behavior aggregate, application flow, or other operationally useful subset of traffic, e.g., routing updates. These MAY include metering, policing, shaping, and packet marking. Traffic conditioning is used to enforce agreements between domains and to condition traffic to receive a differentiated service within a domain by marking packets with the appropriate codepoint in the DS field and by monitoring and altering the temporal characteristics of the aggregate where necessary.

Traffic Conditioner: an entity that performs traffic conditioning functions and which MAY contain meters, policers, shapers, and markers. Traffic conditioners are typically deployed in DS boundary nodes (i.e., not in interior nodes of a DS domain).

Service: a description of the overall treatment of (a subset of) a customer's traffic across a particular domain, across a set of interconnected DS domains, or end-to-end. Service descriptions are covered by administrative policy and services are constructed by applying traffic conditioning to create behavior aggregates which experience a known PHB at each node within the DS domain. Multiple services can be supported by a single per-hop behavior used in concert with a range of traffic conditioners.

Definitions taken from RFC2475 (Differentiated Services Architecture):

- DS boundary node: a DS node that connects one DS domain to a node either in another DS domain or in a domain that is not DS-capable.

- DS-capable: capable of implementing differentiated services as described in this architecture [RFC2475]; usually used in reference to a domain consisting of DS-compliant nodes.

- DS codepoint: a specific value of the DSCP portion of the DS field, used to select a PHB.

- DS-compliant: enabled to support differentiated services functions and behaviors as defined in [DSFIELD], this document (DS Architecture), and other differentiated services documents; usually used in reference to a node or device.

- DS domain: a DS-capable domain; a contiguous set of nodes which operate with a common set of service provisioning policies and PHB definitions.

- DS egress node: a DS boundary node in its role in handling traffic as it leaves a DS domain.

- DS ingress node: a DS boundary node in its role in handling traffic as it enters a DS domain.

- DS interior node: a DS node that is not a DS boundary node.

- Marking: the process of setting the DS codepoint in a packet based on defined rules; pre-marking, re-marking.

- Mechanism: a specific algorithm or operation (e.g., queueing discipline) that is implemented in a node to realize a set of one or more per-hop behaviors.

- Metering: the process of measuring the temporal properties (e.g., rate) of a traffic stream selected by a classifier. The instantaneous state of this process may be used to affect the operation of a marker, shaper, or dropper, and/or may be used for accounting and measurement purposes.

- Policing: the process of discarding packets (by a dropper) within a traffic stream in accordance with the state of a corresponding meter enforcing a traffic profile.

- Service: the overall treatment of a defined subset of a customer's traffic within a DS domain or end-to-end.

- Service Level Agreement: a service contract between a customer and a service provider that specifies the forwarding service a customer should receive. A customer may be a user organization (source domain) or another DS domain (upstream domain). A SLA may include traffic conditioning rules which constitute a TCA in whole or in part.

- Service Provisioning Policy: a policy which defines how traffic conditioners are configured on DS boundary nodes and how traffic streams are mapped to DS behavior aggregates to achieve a range of services.

- Shaping: the process of delaying packets within a traffic stream to cause it to conform to some defined traffic profile.

- Traffic conditioning: control functions performed to enforce rules specified in a TCA, including metering, marking, shaping, and policing.

- Traffic Conditioning Agreement (TCA): an agreement specifying classifier rules and any corresponding traffic profiles and metering, marking, discarding and/or shaping rules which are to apply to the traffic streams selected by the classifier. A TCA encompasses all of the traffic conditioning rules explicitly specified within a SLA along with all of the rules implicit from the relevant service requirements and/or from a DS domain's service provisioning policy.

- Traffic profile: a description of the temporal properties of a traffic stream such as rate and burst size.

- Traffic stream: an administratively significant set of one or more microflows which traverse a path segment. A traffic stream may consist of the set of active microflows which are selected by a particular classifier.

Definitions taken from the Differentiated Services Working Group Charter:

Differentiated Services: The differentiated services approach to providing quality of service in networks employs a small, well-defined set of building blocks from which a variety of services may be built.

Definitions taken from draft-ietf-rap-framework-01.txt:

- Administrative Domain: A collection of network elements under the same administrative control and grouped together for administrative purposes.

- Installed State: A new and unique request made from a PEP to a PDP that must be explicitly deleted.

- Network Element (also called a Node): A networking device, such as a router, a switch, or a hub, where resource allocation decisions have to be made and the decisions have to be enforced.

- QoS Signaling Protocol: A signaling protocol that carries an admission control request for a bandwidth resource, e.g., RSVP.

- Policy: The combination of rules and services where rules define the criteria for resource access and usage.

- Policy control: The application of rules to determine whether or not access to a particular resource should be granted.

- Policy Object: Contains policy-related info such as policy elements and is carried in a request or response related to resource allocation decision.

- Policy Element: Subdivision of policy objects; contains single units of information necessary for the evaluation of policy rules. A single policy element carries an user or application identification whereas another policy element may carry user credentials or credit card information. Examples of policy elements include identity of the requesting user or application, user/app credentials, etc. The policy elements themselves are expected to be independent of which QoS signaling protocol is used.

- Policy Decision Point (PDP): The point where policy decisions are made.

- Policy Enforcement Point (PEP): The point where the policy decisions are actually enforced.

- Policy Ignorant Node (PIN): A network element that does not explicitly support policy control using the mechanisms defined in this document (RAP Framework).

- Resource: Something of value in a network infrastructure to which rules or policy criteria are first applied before access is granted. Examples of resources include the buffers in a router and bandwidth on an interface.

- Service Provider: Controls the network infrastructure and may be responsible for the charging and accounting of services.

- Trusted Node: A node that is within the boundaries of an administrative domain (AD) and is trusted in the sense that the admission control requests from such a node do not necessarily need a PDP decision.

3. Areas of Terminology Conflict

Section 2 listed previously-defined terms that are related to the definition of policy and policy-based networking. Of these terms, the following terms that have been previously defined need more definition in order to satisfy the goals of this document, and will be redefined in subsequent sections of this document:

NAME OF THE TERM	DEFINED IN	PROBLEM WITH DEFINITION
Service	[DIFFSERV], [DIFFARCH]	Tied directly to network traffic; needs to be defined more generally
Policing	[DIFFARCH]	Too specific to traffic conditioning
Service Level Agreement (SLA)	[DIFFARCH]	Too specific to forwarding of network traffic and traffic conditioning
Service Provisioning	[DIFFARCH]	Too specific to Differentiated Services architecture
Policy	[RAPFRAME]	Too simplistic a definition; too focused on network resource control
Policy control	[RAPFRAME]	Too simplistic a definition; too focused on network resource control
Policy Object	[RAPFRAME]	Too specific to using QoS signaling protocol as transport; too focused on network resource control
PDP	[RAPFRAME]	Too RSVP-specific. Needs to add clarification that policy decisions can also be made at the PEP.
Policy Element	[RAPFRAME]	Not just for evaluation of policy Rules
Domain	[DIFFSERV]	Specific to just treatment of Traffic using DiffServ; needs to be generalized

4. Policy Mental Model

In the general sense, policies represent business goals and objectives, and describe how resources are allocated to meet these goals and objectives. In the general sense, a "resource" is any object that can potentially be in short supply, due to many different clients simultaneously requesting it. With respect to the Policy Framework working group, we will differentiate between "resources" as defined above and "network resources", which refers explicitly to resources of a network element (e.g., interface bandwidth). Thus, the term "network resources" matches the definitions of previous documents referenced above.

Again, with respect to networking, policy refers to the ability to administer, manage, and control access to the network resources of network elements in order to provide a set of services to clients of the network. "Clients" in this context refer to users as well as

applications and services.

An underlying assumption of this draft is that policy is stored in a centralized repository. This repository may be, but is not limited to, a directory accessed using the LDAP protocol [LDAP]. The rest of this section defines the underlying mental model to support this definition of policy and policy-based networking.

4.1 General Policy Architecture

A general architecture is shown in Figure 1 below.

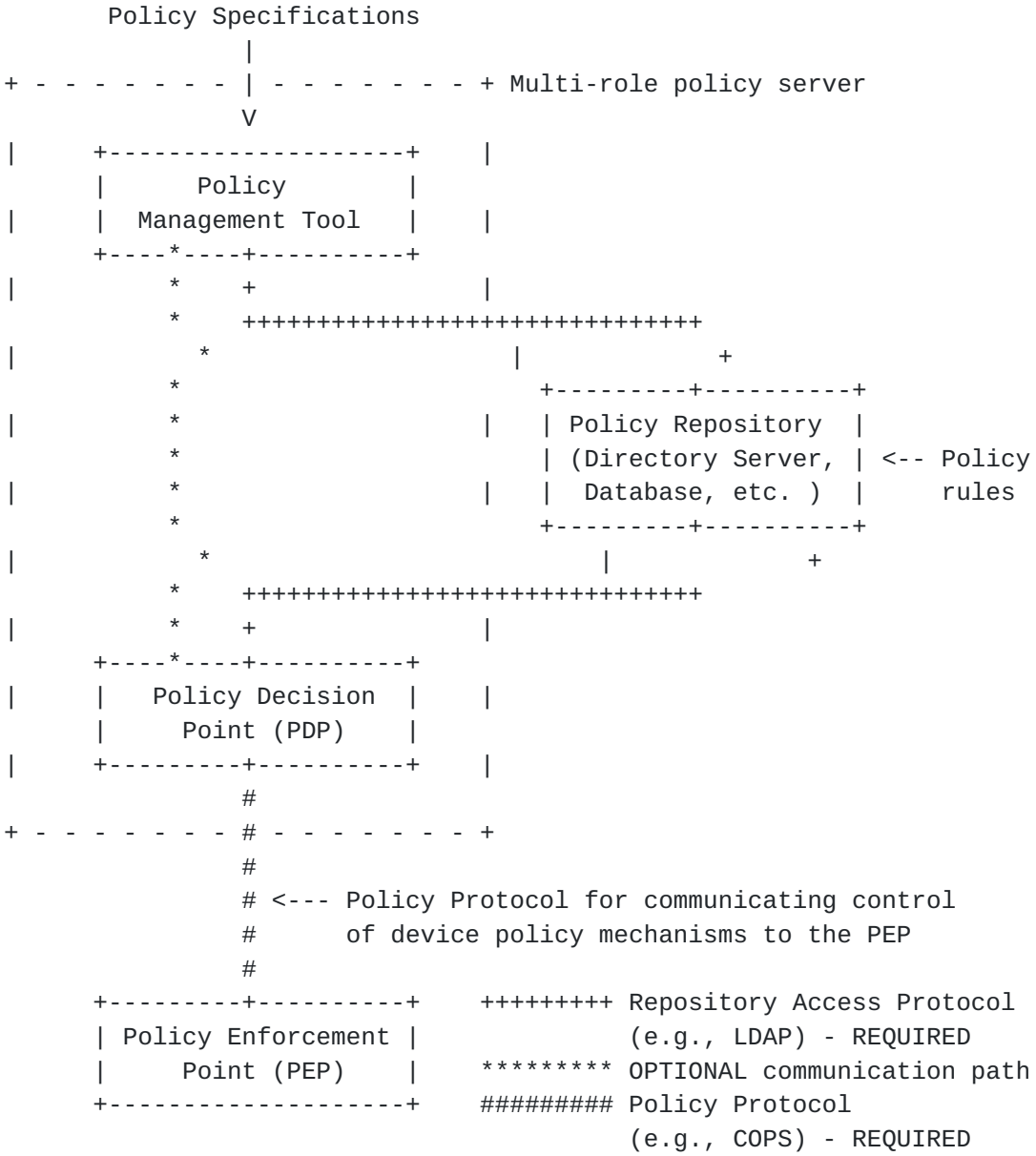

Figure 1. Architectural Overview

The above diagram illustrates one common implementation that combines the use of a policy management tool, a policy repository, a PDP, and a PEP.

This diagram is not meant to imply that these entities must be located in physically separate devices, nor is it meant to imply that the only protocols used for communicating policy are those illustrated. Rather, it is simply meant to show one possible implementation for the purpose of defining the four important entities fundamental to policy: a policy entry/management tool, a policy repository, a PDP, and a PEP. Please refer to [PLYARCH] for a description of how these entities are used and interact with each other.

Note the use of the term "multi-role policy server". This refers to the ability of the policy server to manage and distribute policies of a variety of disciplines, not just those pertaining to networking. However, this document, and the focus of the Policy Framework working group, is limited specifically to networking.

Note also how the policy repository spans the policy server and the rest of the environment. This reinforces thinking of the policy repository as a logically centralized (but possibly physically distributed) repository. It also emphasizes one of the prime goals of the work being done in the Policy Framework working group: interoperability. This is shown explicitly in Figure 2 below.

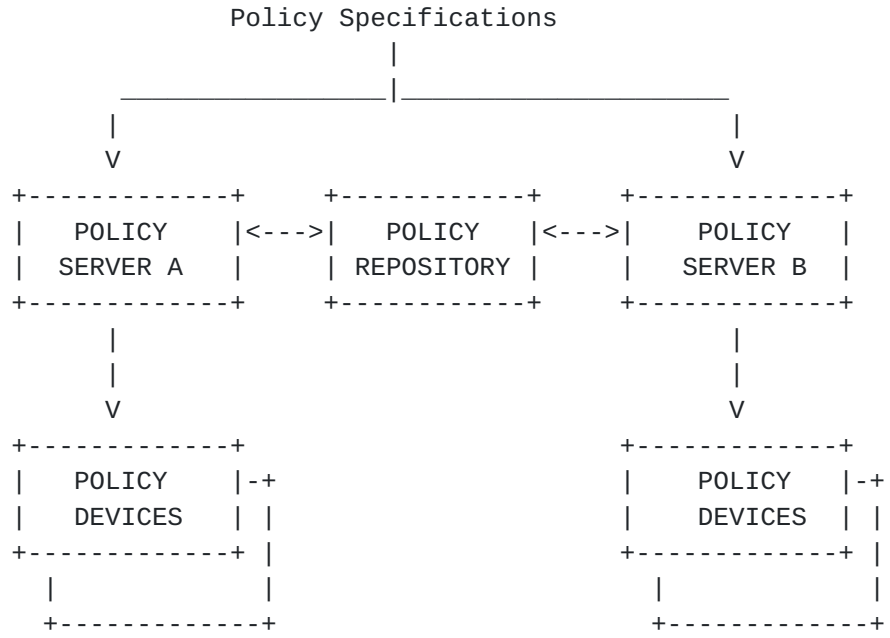

Figure 2. Logically Centralized Policy Repository for Interoperability

Specifically, the above figure provides the flexibility for two different policy servers (A and B) to manage their own devices, while still enabling the policy servers to exchange policy information. The two policy servers could represent different vendors or the same vendor. The latter may be required because of physical location reasons, or

because the types of devices that Policy Server A manages are very different than those managed by Policy Server B (e.g., QoS policy vs. firewall policy).

Although Figure 2 does not show it, the architecture described in Figure 1 also is intended to support the case where a multi-role policy server is not used, and the policy management tool is supplied by one vendor, while the PDP(s) are supplied by another.

Again, the point of interoperability in this case, as well as in the the multiple multi-role policy server case, is the Policy Repository. This is discussed further in [PLYARCH].

It has been previously assumed that all policy decisions will always be made in the PDP and implemented in the PEP. In other words, the PEP is not able to make decisions on its own. This is too simplistic a definition. The problem is that policies that are a function of packet conditions can not be evaluated in the PDP, if the PDP is physically separate from the PEP. This is discussed further in [PLYARCH].

The Repository Access Protocol and the Policy Protocol are in general different protocols. If the Policy Repository is a directory, then LDAP is one example of a Repository Access Protocol. However, the Policy Protocol could be any combination of COPS, SNMP, and Telnet/CLI. In fact, different policy protocols could be used between different devices that are governed by the same PDP.

4.2 How Policy Decisions Are Made

Any PEP that encounters an event requiring a policy-based decision that it can not make by itself first asks its PDP how to handle this request. This results in one or more policy decisions made by the PDP that are in turn communicated to the PEP. A policy decision results in a specific set of operations that either provide a service that was contracted for, or implement a change in state (e.g., in the network) to provide a service. For example, if the network is designed to support differentiated classes of service for handling different types of traffic, network elements could send requests to a policy server asking how to map a given type of traffic, or these network elements could be configured via the policy server or PDP that provides this mapping. Policy decisions would then be made instructing what type of queuing mechanisms to use to handle that traffic (as an example).

4.3 What Is A Policy (In General Terms)

A policy is formally defined as an aggregation of policy rules. Each policy rule is comprised of a set of conditions and a corresponding set of actions. The conditions define when the policy rule is applicable. Once a policy rule is so activated, one or more actions contained by that policy rule may then be executed. These actions are associated with either meeting or not meeting the set of conditions specified in the policy rule.

Policies can contain policies. This notion of hierarchy is crucial, as it enables complex policies to be built from a set of simpler policies, thereby simplifying their management. It also enables reuse of policy building blocks (policy rules, conditions and actions).

This definition of policy can be enhanced in several different ways. However, it is the feelings of the authors that as simple as possible (but no simpler) a definition must first be used and put into practice before more complicated definitions of policy are deployed. Otherwise, it will be much harder to achieve interoperability of policy servers, or other policy entities.

Policy is a relationship among the attributes of the objects maintained by a policy application that control and manages one or more aspects of a set of PEPs. These PEPs are used to provide a set of services that are regulated by one or more policies. Key to this definition is the ability to separate the specification of the set of services to be provided in a vendor-independent manner from the implementation of vendor-specific mechanisms that are applied to vendor-specific network elements to supply those services.

4.4 Real-World Requirements of Policy

There are several important requirements of policy. Principal among these are that it must be:

- able to be used for large-scale distributed systems as well as small point solutions
- widely available to those entities that need access to it
- secure
- easy to detect conflicts

4.4.1 Applicable to Large-Scale Solutions

Nothing in the definition of policy, the architecture that is recommended for managing and distributing policy, or the structure of policy itself should prohibit its applicability to large numbers of objects. By large, we mean potentially millions of objects (e.g., IP phones).

Policy should be able to be put into a form that enables it to be easily distributed to objects in a domain that need it. One way of accomplishing this is to be able to group policy information and objects into policy domains. The existing use of the word domain has been constrained mostly to refer to a set of contiguous network
devices that are under common administrative control. This set of common devices are used to provide a common and consistent set of differentiated services, which are administered in a coordinated fashion.

The term domain, as used in the Policy Framework working group, will be expanded to describe a grouping of not necessarily contiguous devices that are administered in a common way. Here, device is also expanded to encompass hosts, firewalls, and other network resources.

4.4.2 Secure

Security is extremely important for objects that are controlled using policies. This is because limited resources are controlled by policies, and unauthorized access can lead to unfair, or illegal, delegation of and access to these resources. Extra precautions must be taken for managers and policy agents to be authenticated and authorized to perform management actions, in order to protect the system being managed.

4.4.3 Easy to Detect Conflicts

A critical part of the underlying mental model is that it must be easy to detect conflicts between different policies and resolve them. The simplest view of a policy is that it specifies a set of actions that MUST be performed if a set of associated conditions is met. Therefore, the ability to detect and resolve conflicts between policy definitions (conditions as well as actions that are taken when a set of conditions are met) is crucial.

There are many reasons that policies can conflict. A policy can be a member of multiple policy domains, and multiple policies can apply to the same domain. Conflict detection can be done before implementation by syntactic analysis, which will catch the large majority of conflicts. Peculiar cases can occur, such as two policies that appear to conflict but actually don't because they are applied when different situations occur.

One way of doing this is by imposing a priority on both the satisfaction of policy conditions as well as the execution of policy actions. However, it should be noted that priority is a fundamental concepts of distributed systems and MUST be supported irrespective of being used to supply a means for policy conflict resolution.

4.5 Policy Components

Policy is comprised of the following three functions:

1. Decision-making. This compares the current state of the network to a desired state described by an application-specific policy and decides how to achieve or maintain the desired state.
2. Enforcement. This implements a desired policy state through a set of management commands; when applied to network elements, these management commands change the configuration of the device using one or more mechanisms. These mechanisms MAY be vendor-specific.
3. Monitoring: This is an on-going active or passive examination of the network and its constituent devices for checking network health, whether policies are being satisfied, and whether clients are taking unfair advantage of network services.

Decision-making uses static and/or dynamic data to determine if a type of policy is being satisfied and, if not, what steps are needed to satisfy that policy. Enforcement refers to the interpretation and execution of policies by consumers who are assumed to be trustworthy. Policing is the auditing of policy compliance to verify that the policy consumers properly implement policies.

4.6 Policy-Based Applications and Network Policy

Policy is defined in terms of applications or processes that monitor and manipulate one or more entities in order to achieve a desired goal. To make the following discussion simpler to understand, this paper focuses on network policies.

A network policy defines the relationships between clients that desire services of the network and the network elements that provide those services. Network Policy applications model two important things:
1) the state of the SLAs that they are enforcing, and/or
2) some part of the state of the (overall) network in order to ensure that their clients will obtain the services they require of (that portion of) the network.

Such applications will maintain a number of objects of various types, each with one or more attributes. Each of these objects either models the state of one or more network elements or maintains some part of the internal state of the policy application.

These applications contain the mapping of services desired by users to the network elements that provide those services. A network policy, then, is a relationship among attributes of the objects maintained by the policy application that controls and manages some aspect of the network in terms of one or more services that the network provides. Since the application models some part of the state of the entity, it is also accurate to say that a policy is a statement about the desired state of the entity (e.g., the requirements to maintain its current state or the need to transition to a different state).

4.7 Difference Between Policy And Service

Service is a very overloaded word. In CIM, a "service" is used to abstract and manage functionality. Service objects represent the management aspects of the functionality provided by an entity (CIM also defines ServiceAccessPoints, which manage the consumption (or access) of that functionality). The functionality is provided by some other object (e.g., a device and/or software feature object).

The information model of the Policy Framework working group is based on CIM and DEN. A service is therefore the abstraction of managing functionality provided by one or more other objects. It is NOT the functionality itself. Furthermore, a policy is a statement that controls the access to and/or utilization of

resources. A policy can be used to configure and control a service, but it is not the service itself.

4.8 Need For Canonical Representation of Policy

Policies represent business functions and goals. These correspond to network services, which are provided by vendor-specific network elements. The problem is that vendors will describe the same business function in different ways. This is because business functions are described at a necessarily high level, and are therefore subject to different interpretation. Since the network will implement the services that correspond to these business functions, differences in network elements will exacerbate this mapping.

A partial solution is to enforce a consistent representation of policy. All policies MUST consist of a set of policy rules, and all policy rules MUST consist of a set of policy conditions and policy actions. This provides a consistent meta-structure for describing policy, enabling a vendor-independent exchange of policy information.

The policy structure should be amenable to implementing simple policies in a correspondingly lightweight fashion. Hence, simple policy rules are rules whose conditions and actions can be embedded in the policy rule directly. Complex policy rules view the policy rule as a container, in which separate policy condition and/or policy action objects are aggregated.

This structure provides a flexible and extensible representation of policy. However, it does not guarantee interoperability. One possible solution is to use a canonical representation of policy rules, policy conditions and policy actions. This in turn requires a categorization of policy rule, policy condition, and policy action into a set of application-specific domains (e.g., RSVP and Differentiated Services would have separate policy conditions and actions that identify their applicability).

Given a class-based implementation of policy, the above could be implemented easily through defining subclasses that corresponded to the different policies, policy rules, policy conditions, and policy actions that were under control of the PDP. This will be discussed more in [PLYARCH].

5. Policy Terminology

The following is a set of policy terminology definitions.

5.1 Policy Group

A PolicyGroup is a named object that represents an aggregation of PolicyRules. The policy encompassed within a PolicyGroup can describe the overall business function(s) to be accomplished, while the set of policy rules define how those business functions will be met. In this case, the business function(s) correspond to one or more SLOs of an SLA.

Alternatively, the PolicyGroup aggregation could be used to group PolicyRules or other PolicyGroups for the purposes of scoping the jursidiction of the policies, or for the purpose of making the administration of the policies more conveniently accessible to those who are assigned to perform this administration.

5.2 PolicyRule

A PolicyRule is composed of a set of conditions and a corresponding set of actions. This combination in effect defines a sequence of actions to be executed when the corresponding set of conditions is either satisfied or not satisfied. For simplicity, it is recommended that positive (satisfaction) and negative (unable to meet) conditions be realized as separate rules that are folded into a single PolicyRule object.

Each PolicyRule is a declarative statement consisting of a Boolean expression that describes a situation to which the policy applies. In its most general form, when the expression is true, one set of actions is initiated, and when false, a different set of actions is initiated. This version of the document will only consider the expression of a rule as a condition statement (e.g., IF some set of conditions are met, THEN take this set of actions).

5.3 Policy Condition

Policy Conditions consist of two parts, a policy condition type and a policy condition element. This structure is aimed at satisfying the need for a canonical representation of a policy condition.

A policy condition type is a set of predefined conditions that can be attached to a policy rule for evaluation. This canonical set of conditions represent common conditions that all network vendors can implement. By including this canonical representation of policy conditions, the resulting set of policy conditions can be exchanged between multiple vendors' policy servers. This is necessary for interoperability as well as for a policy server to be able to parse rules to determine if any rules conflict may potentially conflict with each other. A policy condition element is a policy condition type instance that is being evaluated. Policy condition elements are related together to form a Boolean expression. The relations can be one of the following operators: in, not in, equal, not equal, less than, less than or equal, greater than, and greater than or equal. (Note: These operators are to be captured in the ExpressionCondition object of the information model current at the time of this writing, as well as in discipline-specific schema subclassed from the core information model that are yet to be defined).

5.4 Policy Action

A policy action is the changing of the configuration of one or more network elements in order to achieve a desired policy state. This (new) policy state provides one or more (new) behaviors. As with policy conditions, a policy action is comprised of two parts, a policy action type and a policy action element. The policy action type defines a canonical set of operations or treatments that can be given to traffic flowing into the network element (e.g., deny, change code point, etc.) that is vendor-independent. Similarly, the policy action element specifies the type of mechanism and/or attribute values to be used to provide the specified operation or treatment.

5.5 Policy Decision

A policy decision is the abstraction of activating and evaluating one or more policy rules. Each policy rule is interpreted in the context of a specific request (implied or explicit) for accessing and/or using one or more resources. It connotes taking one or more pre-determined actions based on whether or not a given set of policy conditions were satisfied.

5.6 Policy Behavior

A policy behavior controls how traffic is treated, what network resources must be utilized, and/or what network services are provided. Policy behaviors define one or more mechanisms that are used to implement the policy. Therefore, different devices can carry out the same policy using different behaviors.

For example, one router might use a dropping behavior and another might use a queuing behavior during times of congestion in order to satisfy an overall delivery goal. Policy behaviors can include one or more of the following:

- permit or deny forwarding of traffic based on:
  - source and destination address
  - source and destination port number
  - protocol type and options
  - other factors specific to vendor implementations, such as host name and/or time of day
- permit or deny access to a requested resource or service
- encrypt the header and/or the payload
- mark or remark the packet
- start or stop accounting and/or auditing
- start or stop logging

5.7 Policy State

A policy state is a description of the settings of one or more network elements. These settings correspond to providing the set of services to be provided by the network. For example, a Service Level Agreement can describe services contracted for by subscribers - this corresponds to a state that various network elements must be put into in order to provide those services.

5.8 Effect of Executing a Policy

A policy determines the behavior of one or more objects that it affects. This behavior takes one of three basic forms:

- allocate (or deny) resources to a given requestor
- add or remove resources to a client already using a service that the policy controls
- allow or prohibit access to a resource

The first two types of policy are contractual policies - they provide resources for a service. The last is an authorization policy - it defines what clients are permitted to do.

5.9 Policy Domain

A policy domain is a collection of objects that have been explicitly grouped together in order to administratively share the same policies. Domains can be nested, in order to reflect hierarchical semantics. Examples are organizational structures, subnets, and a grouping of policies that supply (for example) increasing freedom and/or privileges at lower and lower levels.

A domain does not encapsulate the objects it contains; rather, it holds references to objects that it contains. A domain is thus very similar in concept to a directory or folder in a file system.

Domains can be nested within domains. Note, however, that a nested domain is not necessarily a subset of the parent domain, because an object in the nested domain may not be a direct member of its parent domain.

Policy domains provide a convenient abstraction for specifying policy for individual objects within a large system. Policy domains separate the policy from the entities that the policy affects. This enables the domain membership to be changed without having to change the policy, and vice-versa. It also provides the flexibility to add and remove objects from a policy domain without changing the definition of the policy itself.

5.10 Policy Conflict

A policy conflict occurs when the conditions of two or more policies can be simultaneously satisfied, but the actions of at least one of the policies can not be simultaneously executed. This implies several things:

- one or more policy rules of each of the policies is satisfied by the same request
- each condition of each of the conflicting policy rules is satisfied by the same request
- one or more actions specified by one policy conflict with one or more actions specified by the other policy

Policy conflicts can be resolved in a number of different ways. The simplest is to change the conditions and/or actions of one of the policies so that it no longer conflicts with the other policies. However, if the policies must remain inherently conflicting, then there are a number of ways to resolve the conflict on an individual event basis, including the following:

- apply a "match-first" criteria, wherein conflicts are resolved by matching the first policy that is found
- apply a priority order criteria, wherein conflicts are resolved by finding all policy rules which match a given event and selecting only the rule with the highest priority
- use additional metadata to determine which rule or rules should be applied.

5.10.1 Global Conflict Detection

Global conflict detection refers to the ability to detect conflicts that do not affect any one specific object. For example, two policies both select the same user; one gives him GOLD service and the other gives him Bronze service. Another example: one policy says that engineers get GOLD service, and a second policy says that FTP traffic gets Bronze service; what happens when an engineer uses FTP? Since these types of global conflicts do not affect any one specific object, they can be resolved by a centralized component or by each individual PDP.

5.10.2 Local Conflict Detection

Local conflict detection refers to conflicts that affect specific devices, and/or topology-specific conditions that have conflicting actions. In either case, this type of conflict does affect specific objects. For example, one policy could specify a PHB with 3 queues, while a second policy could specify a PHB with 6 queues. If both of these policies are assigned to the same interface, then there is a local conflict. Note that the global conflict detection component would not have caught this, because it appears that these two

policies do not conflict (one has 3 queues and one has 6 queues). It is only when the policies are applied to the network element that this conflict can be detected.

5.11 Service Level Agreement (SLA)

An SLA is a service contract between a customer and a Service Provider that specifies the expected operational characteristics of their relationship. Example operational characteristics include the details of the treatment which a customer's traffic and/or requests for service should receive. The details of the operational characteristics are defined in terms of Service Level Objectives (SLOs). The SLA documents the agreed levels and parameters of services provided, and can cover a wide range of parameters including items that effect the network element and items that don't (e.g., service hours and availability, user support levels, etc.).

5.12 Service Level Objective (SLO)

An SLO partitions an SLA into individual objectives that can be mapped into policies that can be executed. The SLOs define metrics to enforce, police, and/or monitor the SLA. Some commonly used metrics to determine whether or not an SLA is being fulfilled include component system availability (e.g., up-time and MTBF), performance (e.g., response time), and serviceability (e.g., MTTR).

5.13 Policy Event

A policy event is a notification that triggers one or more policy evaluations. A particular event may or may not initiate a policy decision and/or a policy enforcement action. (Note that events are not explicitly recognized in the schema or framework at the time of this writing. Events can be thought of as implicitly requesting a policy evaluation, when appropriate.)

5.14 Policy Evaluation

Policy evaluation is the determination of whether or not the network (or a part of it) is in a desired policy state. This is usually determined by processing static and/or dynamic data against one or more policy rules, the key point being that the decision is made by comparing definitional data stored in the policy repository with current data from the network that does not have to be stored in the policy repository. If it is found that the network elements are not in the desired policy state, then one or more policy actions will be taken to move the network elements from their current state to the desired state. This is called policy enforcement.

5.15 Policy Enforcement

Policy enforcement is the action of placing the network (or a part of the network) in a desired policy state using a set of management commands. When this definition is applied to network elements, these management commands change the configuration of the device(s) using one or more mechanisms. Enforcement is carried out in the context of a policy rule.

5.16 Policy Monitoring

Policy monitoring is an on-going active or passive examination of the network and its constituent devices for checking network health, whether policies are being satisfied, and whether clients are taking unfair advantage of network services. This is done for one or more of the following reasons:

- to ensure that clients are receiving the services that they have contracted for
- to monitor network statistics as part of checking network health
- to monitor network statistics as part of checking whether policies that are currently in use are being satisfied
- to ensure that clients of a given set of policies are not abusing their privileges

5.17 Policy Agent

A policy agent is a software module that generates and responds to policy events, evaluates policies, and enforces policies.

5.18 Policy-Driven Service

A policy-driven service is a set of cooperating policy agents that define, manage, enforce, and monitor a particular policy.

5.19 Policy Audit

A policy audit examines conditions in one or more active policies to determine if their actions are being executed correctly and the desired result (e.g., flow of traffic) is being carried out. This is equivalent to checking the state of the network to determine the set of policies that are and are not being satisfied.

5.20 Policy Consistency Checking

Policy consistency checking is an analysis of currently active policies to determine their consistency or possible inconsistency with each other. When an inconsistency is discovered, a policy discrepancy is reported.

5.21 Policy Discrepancy

A policy discrepancy is a notification that two or more policies are in actual or potential conflict in the sense that implementation of one policy may prevent or otherwise adversely effect the implementation of the other policies.

5.22 Policy Mechanism

A policy mechanism is a set of vendor-specific commands that configures a network element to put a policy rule into effect.

5.23 Policy Verification

Policy verification is an analysis of a policy to determine if the desired state can be established and maintained for the duration of the corresponding policy rule being active.

5.24 Policy Restoration

Policy Restoration is one or more actions taken to restore the system to a desired policy state. This could be invoked because of a policy action in response to an event, or because the current state got corrupted.

5.25 Roles

In the most general sense, a role describes the duties, rights, and permissions of an object with respect to the rest of the managed environment.

Specifically for the Policy Framework working group, a role is realized via the collection object in the policy information model. This collection object aggregates a network object with other objects that a policy is to be applied to. A role is therefore 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 provides a better means of abstraction that relying on one or more attribute values to group the objects.

For example, the role (collection object, in our info model) "edge interface" can be assigned to the interface of devices to distinguish them for other interfaces, such as "backbone interface", that perform different functions. This enables all devices that have interfaces matching one of these roles to be

referred to by using the descriptive role name, as opposed to having to list a set of IP addresses with their masks.

Roles can be used to identify specific objects (e.g., device interfaces) that should be configured in a common manner using one or more policies. These interfaces may be defined by the purpose that they play in the network (e.g., "edge" vs. "backbone"), the characteristics of the object (e.g., frame relay interfaces require a different configuration than ATM interfaces), or other factors.

"Roles" provide a powerful abstraction mechanism. They enable new policies to be specified for a single role, and have them applied to the devices that use that role. This is much more efficient and less error prone than having to specify a new policy for each and every individual network component. In addition, it enables policies to be modified at the (single) role level, instead of having to search for every occurrence of every policy and individually modify the policy.
But most importantly, it enables the devices and their interfaces to be abstracted from the Policy Server. In other words, the Policy Server no longer needs to have intimate knowledge of each and every device (let alone each and every device interface!) in the network.

At the device (PEP) level, a role is modeled as a collection of interfaces with common interface functions, or collection of objects that contain the common interfaces. In either case, more than one role per object can be defined (see Role Combinations, section 5.25.1, below).

5.25.1 Role Combinations

Role combinations enable an interface to be described by multiple roles. Thus, an object could belong to multiple roles, which is implemented as. multiple collection objects (e.g., "IP Interface", "Classification", and "Edge"). Each of these roles identify particular functions that the object performs in the network, which has its own set of configuration,
or authorization, or other specific functions that are controlled by policy.

6. Policy Example

This section will provide an example of the canonical use of
policies, policy rules, policy conditions and policy actions.
(Note that we are making assumptions here regarding discipline-specific subclasses of PolicyCondition and PolicyAction, for the purposes of this example. At the time of this writing, these discipline-specific subclasses have not yet been defined.)

Assume that the following business rule is to be implemented as
a policy:

> Provide the JitterFreeMPEG2 video service for authorized users between authorized points, but only at agreed-upon times

This rule can be loosely translated as:

```
        IF user IN ApprovedUsers AND service IN VideoServices AND
        source IN VideoSources AND destination IN VideoDestinations
        AND time IN ApprovedTimePeriods
        THEN provide JitterFreeMPEG2
```

The policy condition is loosely translated as:

```
        IF the user is a member of an approved group (ApprovedUsers)
          that are authorized to have this service)
        AND the service requested is one supported (VideoServicesgroup)
        AND the source of the request is approved (in the VideoSources
          group or has been authenticated)
        AND the destination is approved (in the VideoDestinations group
          or has been authenticated)
        AND the time requested is OK (in ApprovedTimePeriods)
```

Here, the policy condition types are:
user, service, source, destination, and time
and the policy condition elements are:
ApprovedUsers, VideoServices, VideoSources, VideoDestinations,
and ApprovedTimePeriods, which are all instances of pre-defined
groups of objects.

The policy action is:

```
    IF the conditions are satisfied
       THEN provide the user with video having a QoS defined by the
            JitterFreeMPEG2 service
```

Note that this policy could require "sub-policies" in order for it to be implemented. For example, RSVP requests might be used to precondition the path between VideoSources and VideoDestinations.

7. Terminology For Implementing Policy GUIs

Part of the task of defining policy terminology is to enable policies to be represented in a user-friendly GUI. This section provides guidelines for doing this through the introduction of additional terminology designed expressly for this purpose.

7.1 Types Of Policies

An effective policy GUI MUST be able to categorize and sort policies in ways that are meaningful to the user. There are three broad means of doing this, based on differentiating between how a policy is used, how a policy is triggered, and the attributes of the policy.

7.2 How A Policy Is Used - Service And Usage Policies

Service and Usage Policies are a means of categorizing policies by what they do and how they are used. Service policies describe services available in the network. Usage policies describe which policies will use which services when the conditions of the usage policies are satisfied. Usage policies describe particular mechanism(s) employed to either maintain the current state of the object, or to transition an object from one state to a new state, in order to utilize the specified services.

For example, the fact that a user can get a particular conditioning treatment, or can use IPSEC to encrypt the payloads of traffic, are both services that are provided and are represented as service policies. On the other hand, differentiating between two flows and assigning different services to the flows is an example of using usage policies to differentiate the handling of the flows.

7.3 Classifying Policies Based On Attributes

Policies can be classified by the attributes that they possess. This includes what the policy applies to (e.g., a class of user or a certain type of application) as well as certain attributes that fundamentally change the applicability (conditions) and effect (actions) of the policy. Examples of these include physical location. For example, if a user is connected to his or her corporate Intranet through the public internet, then a different security policy might be applied to that communication and different restrictions may be placed on accessing resources than when that user connects through the corporate network.

8. Security Considerations

Security and denial of service considerations are not explicitly Considered in this memo, as they are appropriate for the underlying policy architecture and not for its terminology. 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. This terminology draft reinforces the need for policies to be secure.

9. Acknowledegments

Many thanks to the useful discussions and suggestions from the Internet Community at large but especially to Andrea Westerinen, John Lee, Lee Rafalow, Steve Schleimer and Fred Baker.

10. References

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

[DIFFARCH] D. Black, S. Blake, M. Carlson, E. Davies, Z. Whang, W. Weiss, "An Architecture for Differentiated Services", RFC2475, December 1998

[DIFFSERV] K. Nichols and S. Blake, "Definition of the Differentiated Services Field (DS Byte) in the IPv4 and IPv6 Headers", RFC2474, December 1998

[DSFIELD] K. Nichols and S. Blake, "Definition of the Differentiated Services Field (DS Byte) in the IPv4 and IPv6 Headers", Internet Draft <draft-ietf-diffserv-header-00.txt>, May 1998.

[LDAP] This refers to the current collection of LDAP RFCs (2251 - 2256) but especially to RFC 2251 (the protocol).

[RAPFRAME] R. Yavatkar, D. Pendarakis, R. Guerin, "A Framework for Policy-based Admission Control", Internet Draft, <draft-ietf-rap-framework-01.txt>, May 1998

[PLYARCH] G. Waters, et. al., Policy Framework Architecture, Internet Draft, <draft-ietf-policy-arch-00.txt>, June 1999, and subsequent revisions/replacements of this document.

[CIM] CIM Specification and CIM Schemata are available at: http://www.dmtf.org/spec/cims.html

[ISWG] Integrated Services Working Group, whose home page is: http://www.ietf.org/html.charters/intserv-charter.html

[DSWG] Differentiated Services Working Group, whose home page is: http://www.ietf.org/html.charters/diffserv-charter.html

[RAPWG] RSVP Admission Policy Working Group, whose home page is: http://www.ietf.org/html.charters/rap-charter.html

[ISSLLWG] Integrated Services over Specific Link Layers, whose home page is: http://www.ietf.org/html.charters/issll-charter.html

[IPSECWG] IP Security Protocol Working Group, whose home page is: http://www.ietf.org/html.charters/ipsec-charter.html

11. Authors' Addresses

John Strassner
Cisco Systems
Bldg E
190 West Tasman Drive
San Jose, CA 95134
Phone: +1-408-527-1069
Fax: +1-408-527-2477
E-mail: johns@cisco.com

Ed Ellesson
JDGA/501
IBM Corporation
4205 S. Miami Blvd.
Research Triangle Park, NC 27709
Phone: +1-919-254-4115
Fax: +1-919-254-6243
E-mail: ellesson@raleigh.ibm.com

12. Full Copyright Statement