INTERNET-DRAFT
draft-chadha-policy-mpls-te-01.txt
Expires June 2001

K. Isoyama
M. Brunner
M. Yoshida
J. Quittek
NEC Corporation
R. Chadha
G. Mykoniatis
A. Poylisher
R. Vaidyanathan
Telcordia Technologies
A. Kind
IBM
F. Reichmeyer
PfN, Inc.

December 2000

Policy Framework MPLS Information Model for QoS and TE
<draft-chadha-policy-mpls-te-01.txt>

Abstract

The purpose of this draft is to describe an information model for representing MPLS policies, including MPLS for traffic engineering and QoS. RFC 2702, "Requirements for Traffic Engineering over MPLS", is used as a basis for determining the types of information that need to be represented in the traffic engineering part of the information model. The latter document describes the functional capabilities required to implement policies that facilitate efficient and reliable network operations in an MPLS domain.
For the QoS-related part, the Policy Framework QoS Information Model

(QPIM) is extended with new classes for controlling and managing the
lifecycle of Label Switched Paths (LSPs) and for mapping of traffic

onto existing LSPs. The control and management of LSP lifecycles includes mainly the creation, update, and deletion of LSPs and the assignment of QoS properties to LSPs.

This information model may be used by a management system to optimize network performance through the necessary network provisioning actions.

An overview of policy-based management is given in this document, along with a description of the relationship of this work to other information models that are being defined in the IETF Policy Framework working group. A detailed description of the information model and a number of examples illustrating its use follow this.

A UML diagram of this model is available at
http://www.ccrle.nec.de/ietf/MPLS_policy_info_model_01.pdf

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

Table of Contents

1. Introduction

1.1. Goals

The ultimate goal of policy-based networking is to support the trend away from individual device management, toward managing and controlling a network as a whole [PCIM]. Policy-based networking allows network elements from different vendors, equipped with different capabilities, to be consistently configured according to network policies. For instance, network policies may be aligned with the business goals of a company.

MPLS is a combination of switched forwarding with network layer routing. The added value of MPLS is provided by a better price/performance ratio of network layer routing, improved scalability in the network layer, and greater flexibility in the delivery of routing services [MPLSFW]. These advantages are achieved with label switching: a packet belongs to a Forwarding Equivalence Class (FEC). All packets belonging to the same class will get assigned a MPLS label, when it enters the MPLS network. The label in the packet can then be used at subsequent hops between ingress and egress nodes to determine the forwarding treatment by indexing into a table. All packets belonging to a particular FEC and enter the network at the same ingress travel the same path through the network.

Typically, information about bindings of labels to FECs is distributed by a label distribution protocol (e.g. LDP, CR-LDP, BGP, RSVP-TE). The use of policies in the context of MPLS is motivated by the need to provide high-level support for the operation of MPLS networks beyond a standard way of label distribution with LDP or other label distribution protocols.

Some of the important applications of MPLS that are foreseen today are Traffic Engineering (TE), Quality of Service (QoS) support and MPLS based Virtual Private Networks (VPNs).

According to [RFC2702], Traffic Engineering (TE) is concerned with performance optimization of operational networks. In general, it encompasses the application of technology and scientific principles to the measurement, modeling, characterization, and control of Internet traffic, and the application of such knowledge and techniques to achieve specific performance objectives. The aspects of traffic engineering that are of interest for MPLS are measurement and control. A major goal of Internet traffic engineering is to facilitate efficient and reliable network operations while simultaneously optimizing network resource utilization and traffic performance. Traffic engineering has become an indispensable function in many large autonomous systems because of the high cost of network assets and the commercial and competitive nature of the Internet. These factors emphasize the need for maximal operational efficiency.

The concept of MPLS traffic trunks is used extensively in the remainder of this document. According to Li and Rekhter [RFC2430], a traffic trunk is an aggregation of traffic flows of the same class, which are placed inside a Label Switched Path. Essentially, a traffic trunk is an abstract representation of traffic to which specific characteristics can be associated. It is useful to view traffic trunks as objects that can be routed; that is, the path through which a traffic trunk traverses can be changed. In this respect, traffic trunks are similar to virtual circuits in ATM and Frame Relay networks. It is important, however, to emphasize that there is a fundamental distinction between a traffic trunk and the path, and indeed the LSP, through which it traverses. An LSP is a specification of the label switched path through which the traffic traverses.

For QoS support in IP networks, MPLS provides route pinning in order to guarantee certain services to customers. Therefore, high-level means for mapping traffic that matches a specific traffic filter onto an LSP with specific QoS characteristics is needed. Such high-level policies could be used, for instance, with DiffServ over MPLS (MPLSDS) [MPLS-DS].

1.2. Terminology

The following abbreviations are used in this document:

AS	Autonomous System
ATM	Asynchronous Transfer Mode

CIM	Common Information Model
CLI	Command Line Interface
COPS	Common Open Policy Service
DMTF	Distributed Management Task Force
FEC	Forwarding Equivalence Class

LDAP	Lightweight Directory Access Protocol
LSP	Label Switched Path
LSR	Label Switched Router
MAM	Maximum Allocation Multiplier
MIB	Management Information Base
MPLS	Multi-Protocol Label Switching
PCIM	Policy Core Information Model
PDP	Policy Decision Point
QoS	Quality of Service
QPIM	QoS Policy Information Model
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
TE	Traffic Engineering
WG	Work Group

2. Motivation for Policy-based Management of MPLS

The following is a discussion of the advantages of the policy-based management approach for MPLS, which is the approach used in this draft, in comparison to the traditional Internet management approach.

Policy-based management provides the ability to control the network as a whole instead of controlling each individual managed object (network device, interface, queue, LSP) in an independent way. This is also one of the most important advantages of this approach, compared to the traditional network management paradigm, especially in the context of network configuration and provisioning. The reason for this is that traditional managed objects (typically organized by MIB trees) are device-level data models, populated in each router, while the policy information model can be used to represent network-wide entities (e.g. MPLS traffic trunks,) and can be populated in a logically centralized repository. However, note that policy-based management is still possible in a SNMP/MIB based environment. There are various approaches to do so. The policies addressed in this draft are meant to be high-level. The low-level mechanisms for configuration can be implemented with different management protocols.

The object-oriented policy information model defined in this draft enables policy-based management of MPLS networks. The advantages of policy-based management can be realized while performing management tasks, such as MPLS configuration for traffic engineering. In the traditional case, the traffic engineering MIB defined in [MPLS-MIB] provides control of LSP tunnel lifecycles. Consequently, every time the LSP tunnel network design needs to be changed to support new or modified network services, or to react to network events, the MIBs of all the appropriate LSRs have to be modified accordingly.

However, in the policy information model defined in this draft, the administrator can describe changes to the network in terms of policy constructs, thus avoiding the micro-management of the individual LSRs. Also in this case, the policy-based management system may automatically change the network behavior with whatever mechanism available.

The policy-based management approach provides a new level of abstraction that allows network devices from different vendors, supporting different capabilities, to be consistently configured according to high-level network policies.

The selection of the specific policy rules to be applied to each network entity is carried out according to their "role". This concept, introduced in [PCIM], permits the grouping of the network entities according to the role they play in the network.

One very useful concept in this draft is the MPLS traffic trunk [RFC2702]. A traffic trunk represents an aggregation of traffic flows of the same class, placed inside one or more Label Switched Paths. Traffic trunks provide a powerful tool for the administrator or for the automatic traffic engineering module. They can be used to support efficient administration of the way in which traffic is handled by the network.

Another important point is the support of DiffServ, which is already being modeled in a similar fashion [QPIM] and enhanced in order to support DiffServ over MPLS. Using the policy-based management approach will obviously simplify the interaction in the context of configuration and provisioning.

The following example demonstrates some of the points discussed above. A more detailed description of this example can be found in section 6. We are considering a network-wide policy that varies bandwidth allocations based on time of day and type of traffic. Traffic Trunks (TTs) in the network are assigned "roles" in accordance with the Olympic Services Model, i.e. they are marked as "gold", "silver", or "bronze" according to the QoS properties of the traffic that they carry. A network-wide policy controls the bandwidth allocation associated with these traffic trunks.

Specifically, the bandwidth for all gold TTs is reduced by 50% during nights and weekends, which allows traffic in the bronze TTs to be increased during the same time period. This could be potentially useful if the Service Level Agreements are such that a number of customers require gold service during business hours on weekdays, while they subscribe to lower service levels during non-business hours and weekends.

To implement the network-wide policy described above, the following two policy rules are required for the gold TTs:

Policy Rule 1: role is "gold TT"

```
           IF "time is 8am-5pm AND day is Monday to Friday"
           THEN "Modify TT: increase bandwidth by 100%"

   Policy Rule 2: role is "gold TT"
```

```
          IF "(time is 5pm-8am AND day is Monday to Friday) OR
              (day is Saturday to Sunday)"
          THEN "Modify TT: Decrease bandwidth by 50%"
```

The semantics of the first rule is that when the condition becomes true, it increases the bandwidth of all the gold TTs by 100%. In a similar fashion, the semantics of the second rule is that when the condition becomes true, it decreases the bandwidth of all the gold TTs by 50%. Note that this is the same absolute amount.

The role of the two policy rules is set to "gold TT", which indicates which TTs those rules are to be applied to. In this case, these two rules are applied to all the TTs that are characterized as "gold_TT".

The above brief example shows how policies can be applied to MPLS traffic engineering in a network-wide fashion. The above example is explained in detail with reference to our MPLS policy information model in Section 6.

3. Background

The model described in this document is based on the Policy Framework QoS Information Model (QPIM)[QPIM]. QPIM defines classes for representing QoS policy rules, including QoS policy rule conditions and QoS policy rule actions. The classes specified in QPIM address QoS provisioning using Differentiated Services (DiffServ) as well as QoS control via RSVP for Integrated Services (IntServ). QPIM itself refines the Policy Core Information Model (PCIM) [PCIM] which includes generic classes for policy-based networking.

3.1. MPLS concepts and their relation to this model

A general discussion of issues related to MPLS is presented in the framework [MPLS-FW] and architecture [MPLS-ARCH] documents. A brief summary of salient features is provided below as context for the later sections.

3.1.1. Label Switched Paths

MPLS uses the concept of a Label Switched Path (LSP), which is used to carry traffic through the network. An LSP is set up using a signaling protocol and may or may not be associated with QoS guarantees. Currently under discussion are [LDP], [CR-LDP], and [RSVP-TE], where the latter two support setting up LSPs with QoS guarantees. LSPs can be specified in an explicit or implicit manner. Explicitly routed LSPs specify the path that the LSP should take.

Depending on whether the entire sequence of hops is specified or not, the explicit LSP is said to be strictly or loosely routed. Implicitly routed LSPs require a routing mechanism within the network, which decides the path they take.

The LSP is thus a basic object that must be modeled in an MPLS information model. The properties of the LSP depend on what kind of LSPs are taken into account (explicit vs. implicit, with or without QoS etc.). For example, for explicitly routed LSPs, the explicit route for the LSP must also be modeled. Note, however, that the MPLS labels, which carry local significance, are not part of the information modeled about an LSP.

3.1.2. QoS Properties

MPLS in its original fashion has no notion of QoS. However, using MPLS for appropriate traffic engineering enables an ISP to provide a relatively higher quality service to its customers. This improvement in quality is not quantifiable, but is based on the premise that improved network utilization results in improved service quality.

Further, an LSP may be associated with QoS properties, which need to be enforced by QoS mechanisms in LSRs via other, non-MPLS means. In a recent proposal to support DiffServ over MPLS [DS-MPLS] MPLS is used in conjunction with DiffServ for QoS enforcement. This approach allows a Label Switched Router (LSR) to apply a specified Per-Hop Behavior (PHB) to packets of an LSP. In this draft, QoS specification of LSPs and its signaling is covered, but QoS enforcement is not addressed (see [QPIM] instead).

3.1.3. QoS routing for LSPs

The LSP setup process may be constrained by QoS requirements, to achieve a requisite level of service. Such constraint-based routing can be realized in one of two ways. In the first approach, a centralized application, that has knowledge of network-wide QoS state, could calculate the best route through the network for each LSP. The centralized application can then initiate the setup of the LSP, using the explicit route feature of CR-LDP or RSVP-TE to control the path taken by the LSP. In the second approach, the route can be calculated in a distributed fashion, during the signaling process. The MPLS model of this document is transparent to the different QoS routing issues; it only addresses the specification of traffic trunks and LSPs with QoS requirements. The mapping of these requirements to underlying QoS routing mechanisms is out of the scope of this document.

3.1.4. Signaling LSPs

Setting up a new LSP requires signaling or configuration mechanisms.

Two basic mechanisms exist for creating LSPs: (1) using a hop-by-hop signaling protocol like LDP, CR-LDP, or RSVP-TE, or (2) configuring the LSP using SNMP or COPS.

The mechanisms used for LSP setup are transparent to this draft. However, certain CR-LDP-specific mechanisms have been included in this draft (see open issues at the end of this draft). For RSVP traffic profiles, see [QPIM]. The action to be taken from a policy server's point of view is to initiate the setup process with means available in the domain.

3.1.5. Controlling the Signaling Process

The signaling process for setting up a new LSP may be subject to different policies or resource constraints. Therefore, we introduce the mplsPolicyCRLDPSignallingAction as a means to control this process with policies. In some network operation environments this will not be needed, if the policy control is applied before the signaling process in the domain. In others, it is very convenient for the policy-based management system to define policies for the signaling process.

3.1.6. Forwarding Equivalence Classes (FEC)

A Forwarding Equivalence Class (FEC) specifies what traffic is mapped to a specific traffic trunk and/or LSP. The specification of FECs can range from very simple, e.g., just the destination IP address, to very complex, e.g., a set of filters including IP addresses, ports, DSCPs etc. In our model, we associate an FEC with an LSP and/or a traffic trunk. The binding of FECs to LSPs and traffic trunks may be performed during or after the LSP/TT setup.

3.2. Relationship to previous work

3.2.1. Relationship to PCIM

The object-oriented information model described in PCIM provides generic classes for representing policy information. The model allows one to specify network policies in terms of policy rules. A policy rule contains a list of policy conditions and a list of policy actions. Conditions are Boolean expressions that evaluate to either true or false. Actions represent some concrete steps that should be taken by a management system if the conditions in the rule evaluate to true.

The policy framework described in PCIM provides an elaborate mechanism for prioritizing policy rules; specifying time periods during which policy rules are applicable; specifying complex boolean conditions using either disjunctive normal form or conjunctive normal form and negations; grouping policies together; specifying reusable conditions and actions; etc. Policies are specified using a declarative rather than a procedural approach, i.e. policies describe the conditions and actions that make up a rule, but do not give a step-by-step description of how to implement the policy.

The MPLS information model described in this draft refines the notion of policy actions as described in PCIM. New classes are introduced in order to model the following MPLS-related policy actions:

- Generic reservation of a Label Switched Path (LSP)

- Generic setup of Traffic Trunk (TT)
- Reservation of a Label Switched Path (LSP) using CR-LDP
- Policy decisions for a CR-LDP message
- Mapping of traffic into a Label Switched Path (LSP).

Furthermore, new policy classes are defined that inherit from the Policy class in PCIM. The new classes describe:

- Traffic profiles
- Label Switched Paths (LSPs)
- Traffic Trunks
- Forwarding Equivalence Classes (FECs)
- Route Specifications.

3.2.2. Relationship to QPIM

In the QoS Policy Information Model (QPIM) [QPIM], new object classes are defined to model the management and configuration of Diffserv- and RSVP-capable network elements. Apart from QoS-specific information, the model contains a number of concepts that are rather general and can be re-used in a number of different contexts. One such concept is that of variables and constants with well-defined semantics that represent things like IP addresses, port numbers, protocol numbers, etc. These variables are used to describe traffic classifiers in QPIM. Such a representation is also very useful for the information model described in this draft, as there is a need to define the FECs (Forwarding Equivalence Classes) that map to given traffic trunks and LSPs. Other concepts that are re-used from QPIM are those defining traffic profiles and policing actions (see object classes qosPolicyPRTrfcProf and qosPolicyPRAction in [QPIM]). These are used here for describing the traffic profiles of traffic trunks, and the policing requirements for traffic trunks (including what to do with non-conforming traffic for a given traffic trunk).

QPIM lists a set of pre-defined variables that are typically required for layers 2 and 3. This draft extends the list of predefined variables to access the label entries on the label stack of an MPLS packet [MPLS-ENC].

3.2.3. Relationship to MPLS Policy Information Model Requirements

Earlier Internet drafts ([REQPMPLS], [REQMPLS_TE]) discussed policies for MPLS and MPLS-TE. [REQPMPLS] outlines requirements for policy-enabled MPLS and identifies two main categories of MPLS policies: LSP admission policies that map traffic flows onto LSPs, and LSP lifecycle policies that affect LSP creation, deletion, configuration and monitoring. The information model presented in the current draft addresses both these categories of policies, excluding any monitoring requirements. In [REQMPLS_TE], the authors discuss policies for MPLS load balancing. The information model that we present is broader in scope and addresses all aspects of traffic engineering, including load balancing.

The information model in this draft builds upon the PCIM model and further refines it by adding new actions that are specific to the domain of MPLS for traffic engineering and QoS support. Actions are included for creating, modifying, and destroying traffic trunks and LSPs, and assigning LSPs to traffic trunks. Also, new object classes and associations are introduced to model MPLS traffic engineering concepts referenced by these actions, such as traffic trunks, route specifications, LSPs and their hops, resources and resource classes, etc. No attempt is made to define new object classes for representing classifier information as this is currently being addressed in QPIM.

4. Overview of MPLS Policy Information Model

4.1. Overview of object classes and their associations

The following figures show some of the new object classes and associations defined for the MPLS Traffic Engineering and QoS Policy Information Model. A number of object classes from previously defined information models, namely CIM [CIM], are also shown where necessary to illustrate new associations relating newly defined structural object classes to object classes previously defined in those information models. Object classes belonging to these information models are appropriately marked. A detailed description of the depicted object classes is provided in Section 5 of this document.

```
                     +---------------------------+
                     |      FilterList (CIM)     |
                     +-------------^-------------+
                                  *|
                                (a)|
                                  *|
                     +-------------v-------------+
                +---->       mplsPolicyFEC       <----+
             (b)|0..1+---------------------------+0..1|(i)
                |                                     |
                |    +---------------------------+    |
                |    |  qosPolicyTrafficProfile  |    |
                |    +-^-----------------------^-+    |
                |  0..1|                   0..1|      |
                |   (c)|     +------+       (j)|      |       +------+
               *|     *| 0..1|   (d)|         *|     *|      *|   (k)|
     +----------v------v-----v-+    |    +-----v------v-------v-+    |
     |                         <----+    |                      <----+
     |                         |0..1     |                      |*
     | mplsPolicyTrafficTrunk  <--------->    mplsPolicyLSP     |
     |                         |*  (m)  *|                      |
     |                         <--------->                      |
     +--------------------^----+*  (n)  *+--^-------------------+
                         *|                *|
                       (e)|              (l)|
                         *|                *|
                     +----v-----------------v----+
                     |    mplsPolicyRouteSpec    |
                     +----------------------^----+
                                           *|
                                         (f)|                 +------+
                                           *|                *|   (h)|
     +-------------------------+     +------v-----------------v-+    |
     |   ComputerSystem(CIM)   |     |mplsPolicyProtocolEndpoint<----+
     +--------------------^----+     +------^-------------------+*
                         *|                *|
                       (o)|              (g)|
                      0..1|            0..1|
                     +----v-----------------v----+
                     |    mplsPolicyResources    |
                     +---------------------------+
```

Figure 1. Relationships between traffic trunks, LSPs, routes, FECs, traffic profiles hops, resources, etc.

In Figure 1, boxes represent object classes and arrows represent associations between the classes. The following associations appear in Figure 1 above:

(a) mplsPolicyFECFilterSet
(b) mplsPolicyFECofTrunk
(c) mplsPolicyTT_TrafficProfile
(d) mplsPolicyReverseDirTT

(e) mplsPolicyEligibleRouteSpec
(f) mplsPolicyHopInRoute
(g) mplsPolicyEndpointResources
(h) mplsPolicyActiveConnection
(i) mplsPolicyFECofLSP
(j) mplsPolicyLSPTrafficProfile
(k) mplsPolicyLSPinLSP
(l) mplsPolicyRealizes
(m) mplsPolicyCurrentlyAssignedLSP
(n) mplsPolicyBackupLSP
(o) mplsPolicySystemResources

An association represents a relationship between two classes (e.g. associations (a) to (o) excepting (d), (h) and (k) above in Figure 1), or between a class and itself (e.g. associations (d), (h) and (k) in Figure 1). Cardinalities are part of each association; the cardinality of an association indicates how many instances of each class may be related to an instance of the other class. For example, the mplsPolicyBackupLSP association has the cardinality "*" (i.e. "0..n") for both the mplsPolicyTrafficTrunk and the mplsPolicyLSP classes. These ranges are interpreted as follows:

- The "*" written next to mplsPolicyTrafficTrunk indicates that an mplsPolicyLSP may be related to no mplsPolicyTrafficTrunk, to one mplsPolicyTrafficTrunk, or to more than one mplsPolicyTrafficTrunk via the mplsPolicyBackupLSP association. In other words, an LSP may be a backup LSP for zero or more traffic trunks.
- The "*" written next to mplsPolicyLSP indicates that a mplsPolicyTrafficTrunk may be related to no mplsPolicyLSP, to one mplsPolicyLSP, or to more than one mplsPolicyLSP via the mplsPolicyBackupLSP association. In other words, a traffic trunk may have zero or more backup LSPs.

The inheritance tree for the object classes defined in this document is given below. Apart from the new object classes defined in this document, the tree below contains references to object classes defined in the Policy Core Information Model [PCIM], marked "PCIM" below, and to object classes defined in the Common Information Model [CIM], marked "CIM" below. For detailed descriptions of these object classes, see the relevant documents.

```
        +--Policy (abstract) (PCIM)
        |  |
        |  +---PolicyAction (PCIM)
        |  |           |
        |  |           +---mplsPolicyTTAction
        |  |           |
        |  |           +---mplsPolicyLSPAction
        |  |           |   |
        |  |           |   +---mplsPolicyCRLSPResvAction
        |  |           |
        |  |           +---mplsPolicyTT_LSP_Action
        |  |           |
        |  |           +---mplsPolicySignalCtrlAction
        |  |
        |  +---qosPolicyTrfcProf (PQIM)
        |  |           |
        |  |           +--- mplsPolicyCRLSPTrfcProf
        |  |
        |  +---mplsPolicyFEC
        |
        +--CIM_ManagedSystemElement (abstract) (CIM)
           |
           +--CIM_LogicalElement (abstract) (CIM)
              |
              +--mplsPolicyResources
              |
              +--mplsPolicyTrafficTrunk
              |
              +--mplsPolicyLSP
              |
              +--mplsPolicyRouteSpec
              |
              +--CIM_ServiceAccessPoint (abstract) (CIM)
                 |
                 +--CIM_ProtocolEndpoint (abstract) (CIM)
                    |
                    +--mplsPolicyProtocolEndpoint
```

Figure 2. Inheritance Hierarchy for MPLS Policy object classes

In CIM, associations are also modeled as classes. For the MPLS Traffic Engineering Policy Information Model, the inheritance hierarchy is shown below:

```
   [unrooted]
         |
         +---mplsPolicyHopInRoute
         |
         +---mplsPolicySystemResources
         |
         +---mplsPolicyEndpointResources
         |
         +---ActiveConnection (CIM)
         |            |
         |            +---mplsPolicyActiveConnection
         |
         +---mplsPolicyReverseDirTT
         |
         +---mplsPolicyEligibleRouteSpec
         |
         +---mplsPolicyCurrentlyAssignedLSP
         |
         +---mplsPolicyBackupLSP
         |
         +---mplsPolicyRealizes
         |
         +---mplsPolicyLSPinLSP
         |
         +---mplsPolicyTT_TrafficProfile
         |
         +---mplsPolicyLSPTrafficProfile
         |
         +---mplsPolicyFECofTrunk
         |
         +---mplsPolicyFECofLSP
```

Figure 3. Inheritance Hierarchy for associations

4.2. High-level description of information model

Policies in the MPLS domain mainly center on

- lifecycle management of LSPs, including signaling, resource control, admission control etc.,
- management of traffic trunks, including the specification of the traffic traversing the trunks, and
- the mapping of traffic trunks to LSPs, including mapping of DiffServ traffic and tunneling of MPLS traffic over parallel LSPs for the purpose of traffic engineering.

4.2.1. MPLS Signaling Actions

Signaling in MPLS consists of setting up, releasing and updating an LSP along a number of LSRs. It should be possible to control the initiation and the execution of these processes using policies. For some signaling protocols, e.g. the Label Distribution Protocol (LDP), the signaling is initiated by an LSR itself, without any intervention by the Policy Server. The signaling actions in the draft are applicable to constraint based LSP setup using protocols such as CR-LDP and RSVP-TE, and include policy actions to initiate the setup, update, or release of an LSP (mplsPolicyLSPAction), as well as a policy action controlling the LSP setup process (mplsPolicyCRLDPSignalCtrlAction).

4.2.2. MPLS Traffic Engineering Policy Actions

A number of policy actions are defined for the purpose of enabling a management system to manipulate traffic trunks and LSPs. These actions may reference instances of traffic trunks, LSPs, and route specifications (see definition of route specification in a later section). In order to reference such instances, they must first be created and populated; and any related objects and associations must also be instantiated and populated.

For example, mplsPolicyTTAction operates upon a traffic trunk; the property mpTrafficTrunk in this action references an instance of a traffic trunk that is to be operated upon. Thus before instantiating an instance of mplsPolicyTTAction, one must instantiate an instance of mplsPolicyTrafficTrunk that will be referenced by this action. In addition, all the related object instances and associations must also be instantiated before instantiating this action. RFC 2702 [RFC2702] describes the difference between establishing a traffic trunk (which we model by creating an instance of mplsPolicyTrafficTrunk and related classes as described later) and activating a traffic trunk (which is modeled by the mplsPolicyTTAction).

Referencing traffic trunks

The following objects may need to be instantiated in order to fully describe a traffic trunk referenced by an action:

- Route specifications: Zero or more route specifications (mplsPolicyRouteSpec) can be associated with a traffic trunk to indicate that this is a potential route to which this traffic trunk can be mapped. The association is modeled by the mplsPolicyEligibleRouteSpec.

- Traffic profile: A traffic profile (qosPolicyPRTrfcProf) describing the resource requirements of a traffic trunk can be defined for every traffic trunk and associated with the traffic trunk via the mplsPolicyTT_TrafficProfile association.

- Assigned LSPs: Any LSPs (mplsPolicyLSP) currently assigned to a traffic trunk can be defined and associated with it via the mplsPolicyCurrentlyAssignedLSP association.

- Backup LSPs: Any LSPs (mplsPolicyLSP) that are acting as backup LSPs for a traffic trunk can be defined and associated with it via the mplsPolicyBackupLSP association.

- Reverse traffic trunk: If a traffic trunk carrying traffic in the reverse direction exists, it can be associated with a given traffic trunk via mplsPolicyReverseDirTT.

Referencing LSPs

The following objects may need to be instantiated in order to fully describe an LSP referenced by an action:

- Route specifications: Zero or more route specifications (mplsPolicyRouteSpec) can be associated with an LSP to indicate that this LSP is a realization of the route specification. The association is modeled by the mplsPolicyRealizes.

- Traffic trunks: Any traffic trunks (mplsPolicyTrafficTrunk) whose traffic is currently being carried by an LSP should be defined and associated with this LSP via the mplsPolicyCurrentlyAssignedLSP association. Further, any traffic trunks for which an LSP is a backup LSP should be associated with this LSP via the mplsPolicyBackupLSP association.

- LSP hierarchy: If a hierarchy of LSPs exists, an LSP should be associated with any LSPs contained in or containing it via the mplsPolicyLSPinLSP association.

Referencing route specifications

The following objects may need to be instantiated in order to fully describe a route specification referenced by an action:

- LSPs: Zero or more LSPs (mplsPolicyLSP) can be associated with a route specification to indicate that these LSPs are realizations of the route specification. The association is modeled by the mplsPolicyRealizes.

- Traffic trunks: Traffic trunks (mplsPolicyTrafficTrunk) for which a given route specification is a potential route should be associated with the route specification via the mplsPolicyEligibleRouteSpec association.

- Route hops: Every hop specified for a route specification is represented by an instance of mplsPolicyProtocolEndpoint (derived from ProtocolEndpoint in [CIM]) and is associated with a route specification via the mplsPolicyHopInRoute association.

Traffic trunks, LSPs, and supporting object classes and associations

The following object classes and associations are defined. Many of these are referenced by the actions described in the previous section.

- Traffic trunk (object class mplsPolicyTrafficTrunk): For a description of a traffic trunk, see the Introduction section of this document, or see RFC 2702 [RFC2702] for more details. The attributes of a traffic trunk are described by the properties of this object class. A traffic trunk can be associated with LSPs that are currently carrying its traffic (mplsPolicyCurrentlyAssignedLSP association) and with backup LSPs that are used for backup in fault/congestion scenarios (mplsPolicyBackupLSP association). Eligible routes for this traffic trunk are associated with it via the mplsPolicyEligibleRouteSpec association. If a traffic trunk going in the reverse direction exists, it is associated with this traffic trunk via the mplsPolicyReverseDirTT association. Finally, a traffic profile for this traffic trunk can be represented by the qosPolicyPRTrfcProf object class (reused from QPIM) and associated with the traffic trunk via the mplsPolicyTrafficProfile association.

- Route Specification (object class mplsPolicyRouteSpec): This is an object class that represents a route specification from point A to point B. The route specification may or may not be realized in the network by an LSP. A route specification is associated with all the hops contained in this route. These hops are interfaces on LSRs (represented by instances of ProtocolEndpoint, which is an object class defined in the CIM Network Model [CIM]). The associations with these hops are realized via the mplsPolicyHopInRoute associations. Note that a route specification could be an incomplete specification of a route; e.g. it could specify three hops for a route which actually requires at least four hops, and leave the job of choosing the missing hop to a signaling protocol that sets up the corresponding LSP. An LSP can be created based on a route specification using the mplsPolicyLSPAction.

The typical use of this object class is for a network operator to be able to specify potential MPLS routes in advance and later instantiate them by creating LSPs that use this specification. A route specification can be associated with a traffic trunk to indicate that this is a potential route to which this traffic trunk can be mapped (mplsPolicyEligibleRouteSpec association).

- Label-Switched Path (LSP) (object class mplsPolicyLSP): This object class represents an LSP. An LSP can be associated with a route specification in order to indicate that this LSP implements this route specification (mplsPolicyRealizes association). An LSP can also be associated with a traffic trunk if it is currently carrying the traffic for this traffic trunk (via the mplsPolicyCurrentlyAssignedLSP association) or if it is a backup LSP for this traffic trunk (via the mplsPolicyBackupLSP association). An LSP can also be associated with another LSP to indicate a hierarchy of LSPs (mplsPolicyLSPinLSP association).

- Resources (object class mplsPolicyResources): This object class represents resources associated with LSRs and interfaces on these LSRs, such as buffer resources, etc. (mplsPolicySystemResources and mplsPolicyEndpointResources associations, respectively).

- Links and their resources (association mplsPolicyActiveConnection): The resources associated with a link (e.g. bandwidth, etc.) are represented by properties of the association mplsPolicyActiveConnection. This association is used to relate two instances of ProtocolEndpoint that currently have an active connection between them.

Apart from these newly defined object classes and associations, the qosPolicyPRAction object class is reused from QPIM for representing policing actions including actions to be taken for non-conforming traffic. Also, object classes qosPolicySimpleCondition, qosPolicyVariable, and qosPolicyValue and their derived classes from QPIM are used for representing classifier information.

4.3. MPLS Policy Variables

The QoS policy information model specifies a set of pre-defined variables to support a set of fundamental QoS terms that are commonly used in policy conditions [QPIM]. In order to specify meaningful policy rules in the MPLS domain, we extend the set of pre-defined variables with MPLS-specific variables. Here we define a preliminary set of low-level concepts; these may need to be extended in updates to this draft.

The following table defines the pre-defined variables for MPLS and their logical bindings.

Variable name	Logical binding
MPLSLabel	The MPLS label of the flow. Compared to the MPLS label in the MPLS shim header field or the combined VPI/VCI in ATM.
MPLSExp	The MPLS Exp field of the flow. Compared to the MPLS Experimental field in the MPLS shim header.

Table 1: Pre-defined Variable Names and their Bindings

Both variables can be of class type qosPolicyIntegerValue or qosPolicyBitStringValue.

5. Class Definitions

5.1. Class mplsPolicyTTAction

This class represents a policy action that creates, destroys or modifies an instance of a traffic trunk by assigning it to a traffic flow (referred to as "activation" of a traffic trunk in [RFC2702]).

Note that a traffic trunk MUST first be established or defined by creating an instance of mplsPolicyTrafficTrunk and related objects/associations (see Section below for details). RFC 2702 describes the difference between establishing a traffic trunk (which we model by creating an instance of mplsPolicyTrafficTrunk and related classes as described in Section below) and activating a traffic trunk (which is modeled by the mplsPolicyTTAction).

The class definition is as follows:

```
   NAME           mplsPolicyTTAction
  DESCRIPTION   A class used to describe a policy action that
                activates, destroys or modifies a traffic trunk.
  DERIVED FROM  policyAction (defined in [PCIM])
  ABSTRACT      False
  PROPERTIES    mpTrafficTrunk,
                mpTTMode
```

5.1.1. The property mpTrafficTrunk

This property is a reference to the instance of mplsPolicyTrafficTrunk that is to be created, destroyed or modified. The property definition is as follows:

```
  NAME          mpTrafficTrunk
  DESCRIPTION   Traffic trunk being created, destroyed or modified
  SYNTAX        Reference to an mplsPolicyTrafficTrunk
```

5.1.2. The Property mpTTMode

The mpTTMode property specifies whether the action is a creation, destruction, modification of a Traffic Trunk action, or a modification of a traffic profile action (see 5.2.)

```
  NAME          mpTTMode
  DESCRIPTION   Mode of the action
  SYNTAX        Integer (ENUM) - {
                       "TTCreate"=0;
                       "TTModify"=1;
                       "TTDestroy"=2;
                      }
```

5.2. Class mplsPolicy_TT_LSPAction

This class is used to represent the action of assigning or de-assigning an LSP to a traffic trunk. Note that the traffic trunk and LSP are assumed to have been defined by creating instances of mplsPolicyTrafficTrunk and mplsPolicyLSP, respectively, as well as related objects/associations (see Sections X.2.1.1 and X.2.1.2 for details).

```
    NAME           mplsPolicy_TT_LSPAction
    DESCRIPTION    A class that represents an action that
                   assigns or deassigns an LSP to a traffic trunk.
    DERIVED FROM   PolicyAction
    ABSTRACT       False
    PROPERTIES     mpTrafficTrunk,
                  mpAssignedLSP,
                  mpTT_LSPMode
```

5.2.1.
The property mpTrafficTrunk

This property contains a reference to an instance of mplsPolicyTrafficTrunk. Note that the instance of mplsPolicyTrafficTrunk that is referenced can be re-used as it can be referenced by multiple policy actions.

The property definition is as follows:

```
    NAME           mpTrafficTrunk
    DESCRIPTION    Traffic trunk to which an LSP is being
                   assigned
    SYNTAX         Reference to an mplsPolicyTrafficTrunk
```

5.2.2.
The property mpAssignedLSP

This property is a reference to an instance of mplsPolicyLSP. The semantics here are that the traffic trunk referenced within this action is to be sent over the referenced LSP.

The property definition is as follows:

```
    NAME           mpAssignedLSP
    DESCRIPTION    Reference to LSP being assigned to a traffic
                   trunk
    SYNTAX         Reference to an instance of mplsPolicyLSP
```

5.2.3.
The Property mpTT_LSPMode

The mpTT_LSPMode property specifies whether the action is an assignment or deassignment of a LSP to a Traffic Trunk.

```
    NAME           mpTT_LSPMode
    DESCRIPTION    Mode of the action
    SYNTAX         Integer (ENUM) - {
                          "Assign"=0;
                          "Deassign"=1;
                         }
```

5.3. Class mplsPolicyLSPAction

This class defines a generic LSP reservation action. The action initiates the setup, update, or release of an LSP referred to by the in the mpCreateLSP property. The class definition is as follows:

```
 NAME           mplsPolicyLSPAction
 DESCRIPTION    A class used to describe a policy action that
                sets up, releases, or modifies any a LSP.
 DERIVED FROM   PolicyAction (defined in [PCIM])
 ABSTRACT       False
```

```
    PROPERTIES     mpCreateLSP,
                   mpLSPMode
```

5.3.1.
       The Property mpCreateLSP

This property is a reference to an mplsPolicyLSP object, which defines an LSP. The property is defined as follows:

```
    NAME           mpCreateLSP
    DESCRIPTION    LSP being set up, released, or modified
    SYNTAX         Reference to an mplsPolicyLSP object
```

5.3.2.
       The Property mpLSPMode

The mpLSPMode property specifies whether the action is a setup, release, or update action of an LSP.

```
    NAME           mpLSPMode
    DESCRIPTION    Mode of the action
    SYNTAX         Integer (ENUM) - {
                          "LSPSetup"=0;
                          "LSPUpdate"=1;
                          "LSPRelease"=2;
                   }
```

5.4. Class mplsPolicyCRLSPResvAction

This class defines an CR-LSP reservation action using CR-LDP Label Request Messages. The set priority property specifies whether an existing LSP may be preempted in order to The property is mapped to the SetPrio field of the Preemption TLV [CRLDP]. Negotiable properties specify whether the corresponding traffic parameter is negotiable or not. The properties are mapped to the flags field of the Traffic Parameter TLV of [CRLDP].

This class defines a protocol-specific action and will likely be moved to a protocol-specific information model in the future.

The class definition is as follows:

```
    NAME           mplsPolicyCRLSPResvAction
    DESCRIPTION    A class used to describe a policy action that
                   sets up, releases or modifies an LSP using CR-LDP.
    DERIVED FROM   mplsPolicyLSPAction
    ABSTRACT       False
    PROPERTIES     mpCRLSPsetPrio,
                   mpPDRNegotiable, mpPBSNegotiable,
```

                  mpCDRNegotiable, mpCBSNegotiable,
                  mpEBSNegotiable, mpWeightNegotiable

5.4.1. The Property mpCRLSPSetPrio

This property represents the setting priority of the CR-LSP. The property is defined as follows:

```
NAME           mpCRLSPSetPrio
DESCRIPTION    setting priority of the CR-LSP
SYNTAX         Integer (MUST be in the range 0-255)
```

5.4.2. The Property mpPDRNegotiable

This property indicates whether the Peak Data Rate of the CR-LSP is negotiatble or not. The property is defined as follows:

```
NAME           mpPDRNegotiable
DESCRIPTION    Flag that indicates whether PDR is negotiable or not
SYNTAX         Boolean - {"NotNegotiable"=0; "Negotiable"=1}
```

5.4.3. The Property mpPBSNegotiable

This property indicates whether the Peak Burst Size of the CR-LSP is negotiable or not. The property is defined as follows:

```
NAME           mpPBSNegotiable
DESCRIPTION    Flag that indicates whether PBS is negotiable or not
SYNTAX         Boolean - {"NotNegotiable"=0; "Negotiable"=1}
```

5.4.4. The Property mpCDRNegotiable

This property indicates whether the Committed Data Rate of the CR-LSP is negotiable or not. The property is defined as follows:

```
NAME           mpCDRNegotiable
DESCRIPTION    Flag that indicates whether CDR is negotiable or not
SYNTAX         Boolean - {"NotNegotiable"=0; "Negotiable"=1}
```

5.4.5. The Property mpCBSNegotiable

This property indicates whether the Committed Burst Size of the CR-LSP is negotiable or not. The property is defined as follows:

```
NAME           mpCBSNegotiable
DESCRIPTION    Flag that indicates whether CBS is negotiable or not
SYNTAX         Boolean - {"NotNegotiable"=0; "Negotiable"=1}
```

5.4.6. The Property mpEBSNegotiable

This property indicates whether the Excess Burst Size of the CR-LSP is negotiable or not. The property is defined as follows:

NAME	mpEBSNegotiable
DESCRIPTION	Flag that indicates whether EBS is negotiable or not
SYNTAX	Boolean - {"NotNegotiable"=0; "Negotiable"=1}

5.4.7. The Property mpWeightNegotiable

This property indicates whether the Weight of the CR-LSP is negotiable or not. The property is defined as follows:

```
NAME           mpWeightNegotiable
DESCRIPTION    Flag that indicates whether Weight is negotiable or
               not
SYNTAX         Boolean - {"NotNegotiable"=0; "Negotiable"=1}
```

5.5. Class mplsPolicyCRLDPSignalCtrlAction

This class defines a policy action to be applied to CR-LDP messages. The message type property identifies the concerned messages. The mpSendNotification property specifies whether a notification should be sent. If the notification message is sent, the mpErrCode property identifies the error code to be sent. Furthermore, the replace properties specify whether the traffic profile of the CR-LDP message is changed.

This class defines a protocol-specific action and will likely be moved to a protocol-specific information model in the future.

```
NAME           mplsPolicyCRLDPSignalCtrlAction
DESCRIPTION    A class used to describe a policy action that
               is applied to CR-LDP messages.
DERIVED FROM   PolicyAction (defined in [PCIM])
ABSTRACT       False
PROPERTIES     mpCRLDPMessageType, mpSendNotification,
               mpSendRelease, mpErrCode
               mpReplacePDR, mpReplacePBS,
               mpReplaceCDR, mpReplaceCBS,
               mpReplaceEBS, mpReplaceWeight
```

5.5.1. The Property CRLDPMessageType

This property is an enumerated integer and defines different values that limit the scope of the action to be enforced to specific types of CR-LDP Messages. The property is defined as follows:

```
NAME           CRLDPMessageType
DESCRIPTION    A message type to which the action is enforced
SYNTAX         Integer (ENUM) - {
                      "LabelMapping"=0;
                      "LabelRequest"=1;
                      "LabelWithdraw"=2;
                      "LabelRelease"=3;
                      "LabelAbortRequest"=4;
                      "Notification"=5;
```

```
                   }
```

5.5.2. The Property mpSendNotification

This property controls the generation of CR-LDP Notification Messages. The property is defined as follows:

NAME	mpSendNotification
DESCRIPTION	A flag that indicates whether a Notification Message is generated or not.
SYNTAX	Boolean - {No=0, Yes=1}

5.5.3. The Property mpSendRelease

This property controls the generation of CR-LDP Release Messages. The property is defined as follows:

NAME	mpSendRelease
DESCRIPTION	A flag that indicates whether a Release Message is generated or not.
SYNTAX	Boolean - {No=0, Yes=1}

5.5.4. The Property mpErrCode

This property specifies the error code in case the mpSendNotification property has the value 1=yes. If the sent notification property is no, this property is undefined. The error codes are the ones specified in LDP and CR-LDP.

NAME	mpErrCode
DESCRIPTION	Error code of Notification Message
SYNTAX	Integer

5.5.5. The Property mpReplacePDR

This property is a non-negative integer that replaces the Peak Data Rate value in a CR-LDP message. The property is defined as follows:

NAME	mpReplacePDR
DESCRIPTION	Replaced PDR value
SYNTAX	Integer (MUST be non-negative)

5.5.6. The Property mpReplacePBS

This property is a non-negative integer that replaces the Peak Burst Size value in a CR-LDP message. The property is defined as follows:

NAME	mpReplacePBS
DESCRIPTION	Replaced PBS value

```
    SYNTAX        Integer (MUST be non-negative)
```

5.5.7. The Property mpReplaceCDR

This property is a non-negative integer that replaces the Committed Data Rate value in a CR-LDP message. The property is defined as follows:

```
    NAME           mpReplaceCDR
    DESCRIPTION    Replaced CDR value
    SYNTAX         Integer (MUST be non-negative)
```

5.5.8. The Property mpReplaceCBS

This property is a non-negative integer that replaces the Committed Burst Size value in CR-LDP message. The property is defined as follows:

```
    NAME           mpReplaceCBS
    DESCRIPTION    Replaced CBS value
    SYNTAX         Integer (MUST be non-negative)
```

5.5.9. The Property mpReplaceEBS

This property is a non-negative integer that replaces the Excess Burst Size value in a CR-LDP message. The property is defined as follows:

```
    NAME           mpReplaceEBS
    DESCRIPTION    Replaced EBS value
    SYNTAX         Integer (MUST be non-negative)
```

5.5.10. The Property mpReplaceWeight

This property is a non-negative integer that replaces the Weight value in a CR-LDP message. The property is defined as follows:

```
    NAME           mpReplaceWeight
    DESCRIPTION    Replaced Weight value
    SYNTAX         Integer (MUST be in the range 0-255)
```

5.6. Class mplsPolicyLSP

This object class is used to represent an MPLS LSP and its properties. Instances of this class represent existing or to be established LSPs in the network. The class definition is as follows:

```
    NAME              mplsPolicyLSP
    DESCRIPTION       A class with several properties for
                      describing an MPLS LSP.
    DERIVED FROM      LogicalElement
    ABSTRACT          False
    PROPERTIES        mpAdminStatus,
                      mpOperationalStatus,
                      mpLevel,
                      mpLocalLSPID,
                      mpResourceClass,
                      mpHoldingPriority,
                      mpIngressMayReroute,
                      mpIsPersistent,
                      mpIsPinned,
                      mpLocalProtectionAvailable,
                      mpIsAdaptive,
                      Roles
```

5.6.1. The property mpAdminStatus

This property indicates the desired operational status of this LSP.

The property definition is as follows:

```
    NAME           mpAdminStatus
    DESCRIPTION    Administrative status of an LSP.
    SYNTAX         Integer (ENUM) - {
                          "up"=1;
                          "down"=2;
                          "testing"=3;
                   }
```

5.6.2. The property mpOperationalStatus

This property indicates the actual operational status of this LSP, which is typically (but is not limited to) a function of the state of individual segments of this LSP.

The property definition is as follows:

```
    NAME           mpOperationalStatus
    DESCRIPTION    Operational status of an LSP.
    SYNTAX         Integer (ENUM) - {
                          "up"=1;
                          "down"=2;
                          "testing"=3;
                          "unknown"=4;
                          "dormant"=5;
                          "notPresent"=6;
                          "lowerLayerDown"=7
                   }
```

5.6.3. The property mpLevel

This property indicates the nesting level of this LSP.

The property definition is as follows:

```
    NAME           mpLevel
    DESCRIPTION    LSP nesting level.
    SYNTAX        Integer
```

5.6.4. The Property mpLocalLSPId

This property is an integer that indicates the LSP id which is unique with reference to the Ingress LSR.

The property definition is as follows:

NAME	mpLocalLSPID
SYNTAX	Integer (must be in the range 0-65535)

5.6.5. The Property mpResourceClass

This property defines an integer to represent a resource class of the LSP.

The property definition is as follows:

NAME	mpResourceClass
SYNTAX	Integer[] (must be non-negative)

5.6.6. The Property mpHoldingPriority

This property represents the holding priority of the LSP which is already set up. A newly set up LSP is only allowed to preempt the resources of this LSP if its set priority is higher than the hold priority.

NAME	mpHoldingPriority
DESCRIPTION	Holding priority for this LSP
SYNTAX	Integer (must be in the range 0-255)

5.6.7. The property mpIngressMayReroute

This flag indicates that the LSP ingress node may choose to reroute this MPLS route without tearing it down.

The property definition is as follows:

NAME	mpIngressMayReroute
DESCRIPTION	Fast reroute enabled
SYNTAX	boolean

5.6.8. The property mpIsPersistent

This flag indicates whether this LSP should be restored automatically after a failure occurs.

The property definition is as follows:

```
    NAME           mpIsPersistent
    DESCRIPTION    Indicates whether this MPLS route is
                   not
    SYNTAX         boolean
```

5.6.9.
The property mpIsPinned

This flag indicates whether the loosely-routed hops of this MPLS route are to be pinned (see [RSVP-TE][CRLDP]).

The property definition is as follows:

```
NAME           mpIsPinned
DESCRIPTION    Indicates whether the route is pinned
SYNTAX         boolean
```

5.6.10. The property mpLocalProtectionAvailable

This flag permits transit routers to use a local repair mechanism which may result in violation of the explicit routing of this LSP. When a fault is detected on an adjacent downstream link or node, a transit router can reroute traffic for fast service restoration.

The property definition is as follows:

```
NAME           mpLocalProtectionAvailable
DESCRIPTION    Indicates whether local protection is
SYNTAX         boolean
```

5.6.11. The property mpIsAdaptive

An LSP might need to re-route itself (e.g. due to re-optimization, connectivity problems, increase in bandwidth, etc.). If an LSP is configured to be adaptive, it
(a) maintains existing resources until a new path is set up
(b) avoids double-counting for links shared by the old and new paths.

The property definition is as follows:

```
NAME           mpIsAdaptive
DESCRIPTION    A boolean indicating whether an MPLS route is
               adaptive or not.
SYNTAX         boolean
DEFAULT VALUE  true
```

5.6.12. The property Roles

The Roles property specifies the set of roles this LSP may have. This property is defined in the CIM Core Information model [CIM] and therefore its definition is not repeated here. See PCIM for an explanation of how roles are used. (See also open issues at the end of the draft)

5.7. Class mplsPolicyFEC

The mplsPolicyFEC specifies the Forwarding Equivalence Class of an LSP. The FECs may differ depending on the application of MPLS. The association mplsFECFilterSet association associates a FilterList [CIM] with this class.

```
  NAME          mplsPolicyFEC
  DESCRIPTION   A Forwarding Equivalence Class of a Traffic Trunk or
                an LSP
  DERIVED FROM  Policy (defined in [PCIM])
  ABSTRACT      FALSE
  PROPERTIES
```

5.8. Class mplsPolicyCRLSPTrfcProf

This class represents CR-LSP traffic parameters as specified in [CRLDP]. The value of CR-LSP traffic profiles corresponds to the Traffic Parameter TLV carried in CR-LDP messages. The traffic profile is derived from the qosPolicyPRTrfcProf, where the property qpPRExcessBurst is used for the CR-LDP ExcessBurstSize, the qpPRRate replaces CR-LDP PeakDataRate, and qpNormalBurst refers to CR-LDP PeakBurstSize

This class represents protocol-specific parameters and will likely be moved to a protocol-specific information model in the future.

```
  NAME          mplsPolicyCRLSPTrfcProf
  DESCRIPTION   Traffic profile of CR-LSP
  DERIVED FROM  qosPolicyPRTrfcProf (defined in [PQIM])
  ABSTRACT      False
  PROPERTIES    mpCRLSPFrequency,
                mpCRLSPWeight,
                mpCRLSPCommittedDataRate,
                mpCRLSPCommittedBurstSize
```

5.8.1.
The Property mpCRLSPFrequency

This property specifies at what granularity the CDR allocated to the CR-LSP is made available.

```
  NAME          mpCRLSPFrequency
  DESCRIPTION   Granularity of CDR
  SYNTAX        Integer (ENUM){
                        "Unspecified"=0;
                        "Frequent"=1;
                        "VeryFrequent"=2
                        }
```

5.8.2.

The Property mpCRLSPWeight

This property represents the CR-LSP's relative share of the possible excess bandwidth above its committed rate.

```
   NAME          mpCRLSPWeight
```

DESCRIPTION	Relative share of the possible excess bandwidth
SYNTAX	Integer (MUST be in the range 0-255)

5.8.3. The Property mpCRLSPCommittedDataRate

This property is a non-negative integer that defines the token rate parameter of committed data rate token bucket, measured in bytes per second. The property is defined as follows:

NAME	mpCRLSPCommittedDataRate
DESCRIPTION	Committed data rate
SYNTAX	Integer (MUST be non-negative)

5.8.4. The Property mpCRLSPCommittedBurstSize

This property is a non-negative integer that defines the token bucket size parameter of committed data rate token bucket, measured in bytes.

The property is defined as follows:

NAME	mpCRLSPCommittedBurstSize
DESCRIPTION	Committed burst size
SYNTAX	Integer (MUST be non-negative)

5.9. Class mplsPolicyTrafficTrunk

This object class is used to represent an MPLS traffic trunk and its properties. A traffic trunk is an aggregation of traffic flows of the same class which are placed inside an LSP (or more than one LSPs, in the case of load balancing). Essentially, a traffic trunk is an abstract representation of traffic to which specific characteristics can be associated.

This class contains properties describing attributes that can be associated with traffic trunks to influence their behavioral characteristics. Several of these attributes are drawn from RFC 2702 [RFC2702]. The class definition is as follows:

NAME	mplsPolicyTrafficTrunk
DESCRIPTION	A class with several properties for describing an MPLS traffic trunk.
DERIVED FROM	LogicalElement
ABSTRACT	False
PROPERTIES	mpResourceClassAffinity, mpPreemption, mpPriority,

```
                  mpResilience,
                  mpTrafficProportion,
                  mpReoptimizationFreq,
                  Roles
```

5.9.1. The property mpResourceClassAffinity

This property represents the resource class affinity attributes (see [RFC2702]) associated with a traffic trunk which can be used to specify the class of resources that are to be explicitly included or excluded from the path of the traffic trunk (the property mpResourceClass, which appears in object class mplsPolicyResources and in association mplsPolicyActiveConnection, describes the "class" that a resource belongs to). The mpResourceClassAffinity property contains
a list of policy attributes which can be used to impose additional constraints on the path traversed by a given traffic trunk. The resource class affinity property for a traffic trunk contains a string of the form:

```
<resource-class, affinity; resource-class, affinity; ...>
```

The "resource-class" parameter in the above identifies a resource class for which an affinity relationship is defined with respect to the traffic trunk. The "affinity" parameter above is a boolean value that indicates the affinity relationship; that is, whether members of the resource class are to be included or excluded from the path of the traffic trunk. The value "true" signifies explicit inclusion, and the value "false" specifies explicit exclusion.

If no resource class affinity attributes are specified, then a "don't care" affinity relationship is assumed to hold between the traffic trunk and all resources. That is, there is no requirement to explicitly include or exclude any resources from the traffic trunk's path. This should be the default in practice.

Resource class affinity attributes are very useful and powerful constructs because they can be used to implement a variety of policies. For example, they can be used to contain certain traffic trunks within specific topological regions of the network.

A "constraint-based routing" framework (see [RFC2702]) can be used to compute an LSP for a traffic trunk subject to resource class affinity constraints in the following manner:

1. For explicit inclusion, prune all resources not belonging to the specified classes before performing LSP computation.

2. For explicit exclusion, prune all resources belonging to the specified classes before performing LSP computation.

The property definition is as follows:

```
NAME          mpResourceClassAffinity
DESCRIPTION   String containing resource class affinities
SYNTAX        string
```

5.9.2. The property mpPreemption

The preemption attribute (see [RFC2702]) determines whether a traffic trunk can preempt another traffic trunk from a given path, and whether it can be preempted by another traffic trunk. Preemption can used to assure that high priority traffic trunks can always be routed through relatively favorable paths within a differentiated services environment. Preemption can also be used to implement various prioritized restoration policies following fault events.

The preemption property can take one of four values, with the following semantics:

1. preemptor-enabled: can preempt lower priority preemptable traffic trunks
2. non-preemptor: cannot preempt other traffic trunks
3. preemptable: can be preempted by higher priority preemptor-enabled traffic trunks
4. non-preemptable: cannot be preempted.

It is trivial to see that some of the preempt modes are mutually exclusive. Using the numbering scheme depicted above, the feasible preempt mode combinations for a given traffic trunk are as follows: (1, 3), (1, 4), (2, 3), and (2, 4). The (2, 4) combination should be the default.

A traffic trunk, say "A", can preempt another traffic trunk, say "B", only if *all* of the following five conditions hold:

1. "A" has a relatively higher priority than "B"
2. "A" contends for a resource utilized by "B"
3. The resource cannot concurrently accommodate "A" and "B" based on certain decision criteria
4. "A" is preemptor enabled
5. "B" is preemptable.

Preemption is not considered a mandatory attribute under the current best effort Internet service model, although it may be useful. However, in a differentiated services scenario, the need for preemption becomes more compelling. Moreover, in the emerging optical internetworking architectures, where some protection and restoration functions may be migrated from the optical layer to data network elements (such as gigabit and terabit label switching routers) to reduce costs, preemptive strategies can be used to reduce the restoration time for high priority traffic trunks under fault conditions.

The property definition is as follows:

NAME	mpPreemption
DESCRIPTION	Contains preemptability information
SYNTAX	Integer (MUST be in the range 1-4)

5.9.3. The property mpPriority

The priority of a traffic trunk is described by this property. The priority property defines the relative importance of traffic trunks. If a constraint-based routing framework is used with MPLS, priorities can be used to determine the order in which path selection is done for traffic trunks at connection establishment and under fault scenarios. Priorities are also important in implementations permitting preemption because they can be used to impose a partial order on the set of traffic trunks according to which preemptive policies can be applied. The priority of a traffic trunk, along with its preemptability information (see the description of the mpPreemption property in the previous section), determines when it will preempt and/or be preempted by other traffic trunks.

The property definition is as follows:

```
  NAME          mpPriority
  DESCRIPTION   Priority for this traffic trunk.
  SYNTAX        Integer
```

5.9.4. The property mpResilience

The mpResilience property indicates the recovery procedure to be applied to traffic trunks whose paths are impacted by faults. More specifically, it contains a boolean value that determines whether the target traffic trunk is to be rerouted or not when segments of its path fail.

Note that RFC 2702[RFC2702] discusses extended resilience attributes that could be used to specify detailed actions to be taken when faults occur. The representation of such attributes is left for further study.

The property definition is as follows:

```
  NAME          mpResilience
  DESCRIPTION   If set to true, this traffic trunk should be
                rerouted in case of failure; if false, it
                should not.
  SYNTAX        boolean
```

5.9.5. The property mpTrafficProportion

This property is used to indicate the relative proportion of traffic to be carried by parallel traffic trunks. This enables one to perform load distribution across multiple parallel traffic trunks

between two nodes. In many practical situations, the aggregate
traffic between two nodes may be such that no single link can carry
the load. In this case, the only feasible solution is to
appropriately divide the aggregate traffic into sub-streams and
route the sub-streams through multiple paths between the two nodes.
This problem can be addressed by instantiating multiple traffic

trunks between the two nodes, such that each traffic trunk carries a proportion of the aggregate traffic. The proportion of traffic carried by each such trunk is specified by the mpTrafficProportion property.

The property definition is as follows:

```
NAME          mpTrafficProportion
DESCRIPTION   Proportion of traffic to be carried by this
              traffic trunk, specified as a percentage from
              0 to 100.
SYNTAX        Integer
```

5.9.6. The property mpReoptimizationFreq

Due to changes in network and traffic characteristics, there may be a need to periodically change the paths of traffic trunks for optimization purposes. This should not be done too frequently as this could adversely affect the stability of the network. This property indicates how often such reoptimization should be performed.

The property definition is as follows:

```
NAME          mpReoptimizationFreq
DESCRIPTION   Indicates how frequently reoptimization should
              be performed for this traffic trunk. If the
              value of this property is set to zero, this
              indicates that reoptimization should not be
              performed.
SYNTAX        Integer
```

5.9.7. The property Roles

The Roles property specifies the set of roles this TT may have. This property is defined in the CIM Core Information model [CIM] and therefore its definition is not repeated here. See PCIM for an explanation of how roles are used.

5.10. Class mplsPolicyRouteSpec

This object class is used to represent a specification of a path for routing an MPLS traffic trunk/LSP. An LSP can be created based on a route specification using the mplsPolicyLSPAction.

The class definition is as follows:

NAME	mplsPolicyRouteSpec
DESCRIPTION	A class describing an MPLS route specification.
DERIVED FROM	LogicalElement
ABSTRACT	False
PROPERTIES	mpIngressIPAddress, MpEgressIPAddress

5.10.1. The property mpIngressIPAddress

Ingress IP address for this MPLS route.

The property definition is as follows:

NAME	mpIngressIPAddress
DESCRIPTION	Ingress IP address for this MPLS route.
SYNTAX	string

5.10.2. The property mpEgressIPAddress

Egress IP address for this MPLS route.

The property definition is as follows:

NAME	mpEgressIPAddress
DESCRIPTION	Egress IP address for this MPLS route.
SYNTAX	string

5.11. Class mplsPolicyProtocolEndpoint

This class is derived from ProtocolEndpoint [CIM] and represents a hop in the route of LSP or Traffic Trunk. A hop represents an LSR interface and this class provides a way to assign roles of an LSR interface such as "MPLS_INGRESS", "MPLS_EGRESS", "MPLS_CORE", etc.

NAME	mplsPolicyProtocolEndpoint
DESCRIPTION	A class that represents a LSR interface.
DERIVED FROM	ProtocolEndpoint (defined in [CIM])
ABSTRACT	False
PROPERTIES	Roles

5.11.1. The Property Roles

The Roles property specifies the set of roles this mplsPolicyProtocolEndpoint may have. This property is defined in the CIM Core Information model [CIM] and therefore its definition is not repeated here. See PCIM for an explanation of how roles are used.

5.12. Class mplsPolicyResources

This class represents resources associated with LSRs and with interfaces on LSRs. The resources described by this class are associated with the corresponding LSRs/interfaces via the mplsPolicySystemResources and the mplsPolicyEndpointResources associations, respectively.

The class definition is as follows:

```
   NAME          mplsPolicyResources
   DESCRIPTION   Resources associated with LSRs and their
```

```
                    interfaces
   ABSTRACT         False
   DERIVED FROM     LogicalElement,
   PROPERTIES       mpBufferResources,
                    mpMaxAllocMultiplier,
                    mpResourceClass
```

5.12.1. The property mpBufferResources

Buffer resources for an LSR or an LSR interface.

The property definition is as follows:

NAME	mpBufferResources
DESCRIPTION	Buffer resources.
SYNTAX	Integer

5.12.2. The property mpMaxAllocMultiplier

The maximum allocation multiplier (MAM) (see [RFC2702]) of a resource determines the proportion of the resource that is available for allocation to traffic trunks. This attribute is applicable to buffer resources on LSRs. The value of the MAM can be chosen so that a resource can be under-allocated or over-allocated. A resource is said to be under-allocated if the aggregate demands of all traffic trunks that can be allocated to it are always less than the capacity of the resource. A resource is said to be over-allocated if the aggregate demands of all traffic trunks allocated to it can exceed the capacity of the resource.

The property definition is as follows:

NAME	mpMaxAllocMultiplier
DESCRIPTION	Proportion of buffer resources that are available for allocation to traffic trunks.
SYNTAX	Integer

5.12.3. The property mpResourceClass

This property describes the "class" that a resource belongs to (see [RFC2702]). Thus a resource class can be viewed as a "color" assigned to a resource such that the set of resources with the same "color" conceptually belongs to the same class. Resource classes can be used to implement a variety of policies. From a Traffic Engineering perspective, they can be used to implement many policies with regard to both traffic and resource oriented performance optimization. For example, resource class attributes can be used to apply uniform policies to a set of resources; specify the relative preference of sets of resources for path placement of traffic trunks; explicitly restrict the placement of traffic trunks to specific subsets of resources; etc. In general, a resource can be assigned more than one resource class attribute. For example, all of the OC-48 links in a given network may be assigned a distinguished resource class attribute. The subsets of OC-48 links which exist within a given domain of the network may be assigned additional resource class attributes in order to implement specific containment policies, or to architect the network in a certain manner.

The property definition is as follows:

```
NAME           mpResourceClass
DESCRIPTION    Resource class(es) that a resource belongs to.
SYNTAX         Integer[]
```

5.13.
Association mplsPolicySystemResources

The association mplsPolicySystemResources associates a set of resources (object class mplsPolicyResources) with an LSR (modeled by an instance of ComputerSystem as defined in the CIM model [CIM]).

The association definition is as follows:

```
NAME           mplsPolicySystemResources
DESCRIPTION    Associates MPLS resources with an LSR.
ABSTRACT       False
PROPERTIES     mpResources[ref MPLSResources [0..1]],
               mpLSR [ref ComputerSystem[0..n]]
```

5.13.1. The reference mpResources

This property contains an object reference to an mplsPolicyResources instance to which zero or one ComputerSystems (representing LSRs) can be associated. The [0..1] cardinality indicates that there may be zero or one instances of mplsPolicyResources associated with any given LSR.

5.13.2. The reference mpLSR

This property contains an object reference to a ComputerSystem (representing an LSR) that is associated with mplsPolicyResources. The [0..n] cardinality indicates that there may be 0, 1, or more than one LSRs associated with any given mplsPolicyResources instance. These LSRs all have the same resource specifications.

5.14.
Association mplsPolicyEndpointResources

The association mplsPolicyEndpointResources associates a set of resources (object class mplsPolicyResources) with an interface of an LSR (modeled by an instance of mplsPolicyProtocolEndpoint).

The association definition is as follows:

NAME	mplsPolicyEndpointResources
DESCRIPTION	Associates MPLS resources with an interface of an LSR.

```
     ABSTRACT      False
     PROPERTIES    mpResources[ref mplsPolicyResources [0..1]],
                   mpPE [ref mplsPolicyProtocolEndpoint [0..n]]
```

5.14.1. The reference mpResources

This property contains an object reference to an mplsPolicyResources instance to which zero or one mplsPolicyProtocolEndpoints (representing interfaces on LSRs) can be associated. The [0..1] cardinality indicates that there may be zero or one instances of mplsPolicyResources associated with any given LSR interface.

5.14.2. The reference mpPE

This property contains an object reference to a mplsPolicyProtocolEndpoint (representing an LSR interface) that is associated with mplsPolicyResources. The [0..n] cardinality indicates that there may be 0, 1, or more than one LSR interfaces associated with any given mplsPolicyResources instance. These LSR interfaces all have the same resource specifications.

5.15.
      Association mplsPolicyActiveConnection

The association mplsPolicyActiveConnection associates a mplsPolicyProtocolEndpoint with another and represents a link between them. Here mplsPolicyProtocolEndpoint represents an LSR interface.

The association definition is as follows:

```
     NAME          mplsPolicyActiveConnection
     DESCRIPTION   Represents a link between two LSR interfaces
     ABSTRACT      false
     DERIVED FROM  ActiveConnection (from [CIM])
     PROPERTIES    mpEndpoint1 [ref mplsPolicyProtocolEndpoint [0..n]],
                   mpEndpoint2 [ref mplsPolicyProtocolEndpoint [0..n]],
                   mpBandwidth,
                   mpMaxAllocMultiplier,
                   mpResourceClass
```

5.15.1. The reference mpEndpoint1

This property contains a reference to a mplsPolicyProtocolEndpoint instance (representing an LSR interface) to which zero or more ProtocolEndpoints (also representing LSR interfaces) can be associated, representing a connection between the mplsPolicyProtocolEndpoints. The [0..n] cardinality indicates that

there may be zero or more instances of mplsPolicyProtocolEndpoint
associated with any given mplsPolicyProtocolEndpoint.

5.15.2. The reference mpEndpoint2

This property contains a reference to a mplsPolicyProtocolEndpoint instance (representing an LSR interface) to which zero or more ProtocolEndpoints (also representing LSR interfaces) can be associated, representing a connection between the mplsPolicyProtocolEndpoints. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyProtocolEndpoint associated with any given mplsPolicyProtocolEndpoint.

5.15.3. The property mpBandwidth

Bandwidth for the link represented by this connection.
The property definition is as follows:

```
NAME           mpBandwidth
DESCRIPTION    Link bandwidth
SYNTAX         Integer
```

5.15.4. The property mpMaxAllocMultiplier

The maximum allocation multiplier (MAM) (see [RFC2702]) of a resource determines the proportion of the resource that is available for allocation to traffic trunks. This attribute is applicable to link bandwidth. The values of the MAM can be chosen so that a resource can be under-allocated or over-allocated. A resource is said to be under-allocated if the aggregate demands of all traffic trunks that can be allocated to it are always less than the capacity of the resource. A resource is said to be over-allocated if the aggregate demands of all traffic trunks allocated to it can exceed the capacity of the resource.

The property definition is as follows:

```
NAME           mpMaxAllocMultiplier
DESCRIPTION    Proportion of link bandwidth that is available
               for allocation.
SYNTAX         Integer
```

5.15.5. The property mpResourceClass

This property describes the "class" that a resource belongs to. Thus a resource class can be viewed as a "color" assigned to a resource such that the set of resources with the same "color" conceptually belong to the same class. Resource classes can be used to implement a variety of policies. From a Traffic Engineering perspective, they can be used to implement many policies with regard to both traffic and resource oriented performance optimization. For example, resource class attributes can be used to apply uniform policies to a set of resources; specify the relative preference of sets of resources for path placement of traffic trunks; explicitly restrict the placement of traffic trunks to specific subsets of resources; etc. In general, a resource can be assigned more than one resource class attribute. For example, all of the OC-48 links in a given network may be assigned a distinguished resource class attribute. The subsets of OC-48 links which exist within a given domain of the network may be assigned additional resource class attributes in order to implement specific containment policies, or to architect the network in a certain manner.

The property definition is as follows:

```
  NAME           mpResourceClass
  DESCRIPTION    Resource class assigned to this link.
  SYNTAX         Integer[]
```

5.16.
      Association mplsPolicyHopInRoute

The association mplsPolicyHopInRoute provides a way to associate hops with mplsPolicyRouteSpec. Hops are instances of mplsPolicyProtocolEndpoint and represent LSR interfaces.

The association definition is as follows:

```
  NAME           mplsPolicyHopInRoute
  DESCRIPTION    Associates hops with mplsPolicyRouteSpec
  ABSTRACT       False
  PROPERTIES     mpRoute[ref mplsPolicyRouteSpec[0..n]],
                 mpHop[ref mplsPolicyProtocolEndpoint[0..n]],
                 mpIsStrict,
                 mpOrder
```

5.16.1. The reference mpRoute

This property contains an object reference to an mplsPolicyRouteSpec with which a number of hops can be associated. The [0..n] cardinality indicates that there may be 0, 1, or more than one mplsPolicyRouteSpec associated with any given hop, indicating that this hop is contained in all these routes.

5.16.2. The reference mpHop

This property contains an object reference to a mplsPolicyProtocolEndpoint (representing an LSR interface) that is a hop for an mplsPolicyRouteSpec. The [0..n] cardinality indicates that there may be 0, 1, or more than one hops associated with any given mplsPolicyRouteSpec. These are all the hops that are included in the route specification.

5.16.3. The property mpIsStrict

Denotes whether the referenced hop is routed in a strict or loose fashion.

The property definition is as follows:

```
     NAME           mpIsStrict
     DESCRIPTION    Denotes whether the referenced hop is routed
                    in a strict or loose fashion.
     SYNTAX         boolean
```

5.16.4. The property mpOrder

This property indicates the hop sequence, 1..n.

The property definition is as follows:

```
     NAME           mpOrder
     DESCRIPTION    Hop sequence
     SYNTAX         Integer
```

5.17. Association mplsPolicyEligibleRouteSpec

The association mplsPolicyEligibleRouteSpec associates an MPLS traffic trunk with a route specification that is a potential route for this traffic trunk.

The association definition is as follows:

```
     NAME           mplsPolicyEligibleRouteSpec
     DESCRIPTION    Associates a traffic trunk with a route
                    specification that is a potential route for
                    this traffic trunk.
     ABSTRACT       false
     PROPERTIES     mpTT [ref MplsPolicyTrafficTrunk [0..n]],
                    mpRoute [ref MplsPolicyRouteSpec [0..n]],
                    mpPreference,
                    mpIsMandatory
```

5.17.1. The reference mpTT

This property contains an object reference to an mplsPolicyTrafficTrunk instance associated with an mplsPolicyRouteSpec. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyTrafficTrunk associated with any given mplsPolicyRouteSpec; i.e. zero or more traffic trunks may share the same route specification, indicating that this route specification is an eligible route for these traffic trunks.

5.17.2. The reference mpRoute

This property contains an object reference to an mplsPolicyRouteSpec that is associated with an mplsPolicyTrafficTrunk. The [0..n] cardinality indicates that there may be 0, 1, or more than one

mplsPolicyRouteSpecs associated with any given
mplsPolicyTrafficTrunk instance; i.e. any traffic trunk can have
multiple eligible route specifications.

5.17.3. The property mpPreference

This property represents the preference for the referenced route by the referenced traffic trunk.

The property definition is as follows:

```
  NAME           mpPreference
  DESCRIPTION    Preference for the referenced route for the
                referenced traffic trunk.
  SYNTAX         Integer
```

5.17.4. The property mpIsMandatory

Indicates whether this is a mandatory route for this traffic trunk or not.

The property definition is as follows:

```
  NAME           mpIsMandatory
  DESCRIPTION    Indicates whether this is a mandatory route
                 for this traffic trunk or not.
  SYNTAX         boolean
```

5.18. Association mplsPolicyReverseDirTT

The association mplsPolicyReverseDirTT associates an MPLS traffic trunk with a traffic trunk going in the reverse direction.

The association definition is as follows:

```
  NAME           mplsPolicyReverseDirTT
  DESCRIPTION    Associates a traffic trunk with another going
                in the reverse direction.
  ABSTRACT       False
  PROPERTIES     mpTT1 [ref mplsPolicyTrafficTrunk [0..1]],
                 mpTT2 [ref mplsPolicyTrafficTrunk [0..1]]
```

5.18.1. The reference mpTT1

This property contains an object reference to an mplsPolicyTrafficTrunk instance to which zero or one mplsPolicyTrafficTrunks can be associated. An mplsPolicyReverseDirTT association between two traffic trunks represents the fact that these traffic trunks carry traffic in opposite directions. The [0..1] cardinality indicates that there may be zero or one instances of mplsPolicyTrafficTrunk associated with any given mplsPolicyTrafficTrunk.

5.18.2. The reference mpTT2

This property contains an object reference to an mplsPolicyTrafficTrunk instance to which zero or one mplsPolicyTrafficTrunks can be associated. An mplsPolicyReverseDirTT association between two traffic trunks represents the fact that these traffic trunks carry traffic in opposite directions. The [0..1] cardinality indicates that there may be zero or one instances of mplsPolicyTrafficTrunk associated with any given mplsPolicyTrafficTrunk.

5.19. Association mplsPolicyRealizes

The association mplsPolicyRealizes associates an LSP with an MPLS route specification (mplsPolicyRouteSpec). Such an association exists if the referenced LSP is an implementation of the referenced route specification. Recall that a route specification could be loosely or strictly specified; an LSP associated to a route specification via the mplsPolicyRealizes association satisfies all the constraints laid down in this route specification.

The association definition is as follows:

```
  NAME           mplsPolicyRealizes
  DESCRIPTION    Associates an LSP with a route specification.
  ABSTRACT       False
  PROPERTIES     mpLSP [ref mplsPolicyLSP [0..n]],
                 mpRouteSpec [ref mplsPolicyRouteSpec [0..n]]
```

5.19.1. The reference mpLSP

This property contains an object reference to an mplsPolicyLSP instance associated with an mplsPolicyRouteSpec. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyLSP associated with any given mplsPolicyRouteSpec; i.e. zero or more LSPs can be a realization of the same route specification.

5.19.2. The reference mpRouteSpec

This property contains an object reference to an mplsPolicyRouteSpec that is associated with an mplsPolicyLSP. The [0..n] cardinality indicates that there may be 0, 1, or more than one mplsPolicyRouteSpecs associated with any given mplsPolicyLSP instance; i.e. any LSP can be a realization of have zero or more route specifications.

5.20. Association mplsPolicyCurrentlyAssignedLSP

The association mplsPolicyCurrentlyAssignedLSP associates the LSP to which a traffic trunk is assigned with that traffic trunk.

The association definition is as follows:

```
NAME           mplsPolicyCurrentlyAssignedLSP
DESCRIPTION    Associates a traffic trunk with the LSP
               assigned to it.
ABSTRACT       False
PROPERTIES     mpTT [ref MplsPolicyTrafficTrunk [0..n]],
               mpLSP [ref mplsPolicyLSP [0..n]]
```

5.20.1. The reference mpTT

This property contains an object reference to an mplsPolicyTrafficTrunk instance to which zero or more mplsPolicyLSPs can be associated. An mplsPolicyCurrentlyAssignedLSP association between a traffic trunk and an LSP represents the fact that this LSP is currently carrying the traffic for this traffic trunk. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyTrafficTrunk associated with any given mplsPolicySLSP, i.e. an LSP could be carrying traffic for zero or more traffic trunks.

5.20.2. The reference mpLSP

This property contains an object reference to an mplsPolicyLSP instance to which zero or more mplsPolicyTrafficTrunks can be associated. An mplsPolicyCurrentlyAssignedLSP association between a traffic trunk and an LSP represents the fact that this LSP is currently carrying the traffic for this traffic trunk. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyLSP associated with any given mplsPolicyTrafficTrunk, i.e. these LSPs are sharing the traffic load for this traffic trunk.

5.21. Association mplsPolicyBackupLSP

The association mplsPolicyBackupLSP associates a backup LSP with an MPLS traffic trunk. The idea is that this LSP can be used as a backup for this traffic trunk if necessary.

The association definition is as follows:

```
NAME           mplsPolicyBackupLSP
DESCRIPTION    Associates a backup LSP with a traffic trunk.
ABSTRACT       False
PROPERTIES     mpTT [ref mplsPolicyTrafficTrunk [0..n]],
               mpLSP [ref mplsPolicyLSP [0..n]],
               mpPreference
```

5.21.1. The reference mpTT

This property contains an object reference to an

mplsPolicyTrafficTrunk instance with which zero or more
mplsPolicyLSPs can be associated. An mplsPolicyBackupLSP association
between a traffic trunk and an LSP represents the fact that this LSP
is a backup for this traffic trunk and can be used in
failure/congestion situations. The [0..n] cardinality indicates that

there may be zero or more instances of mplsPolicyTrafficTrunk associated with any given mplsPolicyLSP, i.e. an LSP can be a backup for zero or more traffic trunks.

5.21.2. The reference mpLSP

This property contains an object reference to an mplsPolicyLSP instance with which zero or more mplsPolicyTrafficTrunks can be associated. An mplsPolicyBackupLSP association between a traffic trunk and an LSP represents the fact that that this LSP is a backup for this traffic trunk and can be used in failure/congestion situations. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyLSP associated with any given mplsPolicyTrafficTrunk, i.e. there may be zero or more backup LSPs for this traffic trunk.

5.21.3. The property mpPreference

This property represents the preference for the referenced backup LSP by the referenced traffic trunk. In other words, an explicit order can be imposed on all the backup LSPs for a traffic trunk to indicate a sequence of backup LSPs ordered from most preferred to least preferred.

The property definition is as follows:

```
  NAME           mpPreference
  DESCRIPTION    Preference for the referenced backup LSP for
                the referenced traffic trunk.
  SYNTAX         Integer
```

5.22.
      Association mplsPolicyLSPinLSP

The association mplsPolicyLSPinLSP associates an LSP with another LSP and indicates a hierarchical relationship between the two LSPs.

The association definition is as follows:

```
  NAME           mplsPolicyLSPinLSP
  DESCRIPTION    Associates an LSP with another LSP, indicating
                 a hierarchical relationship between the two.
  ABSTRACT       False
  PROPERTIES     mpContainingLSP [ref mplsPolicyLSP [0..n]],
                 mpContainedLSP [ref mplsPolicyLSP [0..n]]
```

5.22.1. The reference mpContainingLSP

This property contains an object reference to an mplsPolicyLSP instance associated with another mplsPolicyLSP. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyLSP associated with other mplsPolicyLSPs via this association, indicating that these LSPs all contain the referenced LSP.

5.22.2. The reference mpContainedLSP

This property contains an object reference to an mplsPolicyLSP instance associated with another mplsPolicyLSP. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyLSP associated with other mplsPolicyLSPs via this association, indicating that these LSPs are all contained in the referenced LSP.

5.23.
Association mplsPolicyTT_TrafficProfile

The association mplsPolicyTT_TrafficProfile associates a traffic trunk with a traffic profile, represented by an instance of qosPolicyTrafficProfile. This traffic profile describes the characteristics of the traffic carried by the referenced traffic trunk.

The association definition is as follows:

```
  NAME           mplsPolicyTT_TrafficProfile
  DESCRIPTION    Associates a traffic trunk with a traffic
                 profile.
  ABSTRACT       False
  PROPERTIES     mpTT [ref mplsPolicyTrafficTrunk [0..n]],
                 mpTrfcProf[ref qosPolicyTrafficProfile [0..1]]
```

5.23.1. The reference mpTT

This property contains an object reference to an mplsPolicyTrafficTrunk instance associated with a qosPolicyTrafficProfile. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyTrafficTrunk associated with any given qosPolicyTrafficProfile; i.e. zero or more traffic trunks can share the same traffic profile specification.

5.23.2. The reference mpTrfcProf

This property contains an object reference to a qosPolicyTrafficProfile that is associated with an mplsPolicyTrafficTrunk. The [0..1] cardinality indicates that there may be zero or one traffic profiles associated with any given traffic trunk.

5.24.

Association mplsPolicyLSP_TrafficProfile

The association mplsPolicyLSP_TrafficProfile associates an LSP with a traffic profile, represented by an instance of qosPolicyTrafficProfile. This traffic profile describes the characteristics of the traffic carried by the referenced LSP.

The association definition is as follows:

```
  NAME           mplsPolicyLSP_TrafficProfile
  DESCRIPTION    Associates an LSP with a traffic profile.
  ABSTRACT       False
  PROPERTIES     mpLSP [ref mplsPolicyLSP [0..n]],
                 mpTrfcProf[ref qosPolicyTrafficProfile [0..1]]
```

5.24.1. The reference mpLSP

This property contains an object reference to an mplsPolicyLSP instance associated with a qosPolicyTrafficProfile. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyLSP associated with any given qosPolicyTrafficProfile; i.e. zero or more LSPs can share the same traffic profile specification.

5.24.2. The reference mpTrfcProf

This property contains an object reference to a qosPolicyTrafficProfile that is associated with an mplsPolicyLSP. The [0..1] cardinality indicates that there may be zero or one traffic profiles associated with any given LSP.

5.25. Association mplsPolicyFECofTrunk

The association mplsPolicyFECofTrunk associates a Traffic Trunk with an FEC, represented by an instance of qosPolicyTrafficTrunk.

The association definition is as follows:

```
  NAME           mplsPolicyFECofTrunk
  DESCRIPTION    Associates a Traffic Trunk with a FEC.
  ABSTRACT       False
  PROPERTIES     mpTT [ref mplsPolicyTrafficTrunk [0..n]],
                 mpFEC[ref mplsPolicyFEC [0..1]]
```

5.25.1. The reference mpLSP

This property contains an object reference to an mplsPolicyTrafficTrunk instance associated with a mplsPolicyFEC. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyTrafficTrunk associated with any given mplsPolicyFEC; i.e. zero or more LSPs can share the same FEC specification.

5.25.2. The reference mpFEC

This property contains an object reference to an mplsPolicyFEC that is associated with an mplsPolicyTrafficTrunk. The [0..1] cardinality indicates that there may be zero or one FEC associated with any given traffic trunk.

5.26.
      Association mplsPolicyFECofLSP

The association mplsPolicyFECofLSP associates an LSP with a FEC.

The association definition is as follows:

NAME	mplsPolicyFECofLSP
DESCRIPTION	Associates an LSP with a FEC.
ABSTRACT	False
PROPERTIES	mpLSP [ref mplsPolicyLSP [0..n]], mpFEC[ref mplsPolicyFEC [0..1]]

5.26.1. The reference mpLSP

This property contains an object reference to an mplsPolicyLSP instance associated with a mplsPolicyFEC. The [0..n] cardinality indicates that there may be zero or more instances of mplsPolicyLSP associated with any given mplsPolicyFEC; i.e. zero or more LSPs can share the same FEC specification.

5.26.2. The reference mpFEC

This property contains an object reference to an mplsPolicyFEC that is associated with an mplsPolicyTrafficTrunk. The [0..1] cardinality indicates that there may be zero or one FEC associated with any given LSP.

5.27.
      Association mplsPolicyFECFilterSet

The association mplsPolicyFECFilterSet associates a FilterList [CIM] with an mplsPolicyFEC. A FilterList represents the packet identifier of the FEC.

NAME	mplsPolicyFECFilterSet
DESCRIPTION	
ABSTRACT	False
PROPERTIES	mpFEC[ref mplsPolicyFEC[0..n], mpFilter[ref FilterList[0..n], mpFilterListPosition

5.27.1. The Property mpFEC

This property contains an object reference to an mplsPolicyFEC instance with which a number of filters can be associated. The [0..n] cardinality indicates that there may be 0, 1, or more than one

mplsPolicyFEC associated with any given filter. i.e. zero or more
FECs can share the same Filter specification.

5.27.2. The Property mpFilter

This property contains an object reference to a FilterList[CIM] instance. The [0..n] cardinality indicates that there may be 0, 1, or more than one FilterList associated with any given FEC.

5.27.3. The Property mpFilterListPosition

This property indicates the position of the FilterList in the FEC. The property is defined as follows:

```
NAME           mpFilterListPosition
DESCRIPTION
SYNTAX         Integer (MUST be non-negative)
```

6. Examples

6.1. Establishing an LSP

This section provides an example that shows how the classes defined above as part of the QoS information model for MPLS may be used in a typical policy scenario. For the example scenario we assume an MPLS network and two hosts (A and B) outside the MPLS domain (see Figure 4).

```
       1.2.3.4
        +--------+
        | Host A |         +------------------+
        +--------+        /                    \
            |         +---------+              |  2.3.4.5
            \--...--->| Ingress |         +--------+
                      +---------+         | Egress |--\
                1.2.3.5   |               +--------+  |
                          \   MPLS domain      /      |      +--------+
                           +------------------+       \-...->| Host B |
                                                             +--------+
                                                            2.3.4.6
```

Figure 4: Example MPLS Network

Furthermore, we assume for the scenario that traffic from A to B should be mapped to a specific forwarding equivalence class of a newly created LSP. The LSP be specified with a committed data rate of 10Mb/s.

A provisioning policy to establish the LSP can be represented using the classes defined in the MPLS information model as follows:

```
       +--------------+
       |  PolicyRule  |
       +--------------+
         |
         |
       +------------------------+
       |  mplsPolicyLSPAction   |
       +------------------------+
         |
         |
       +------------------+
       |  mplsPolicyLSP   |
       +------------------+
         |
         |   +-----------------+
         |-- |  mplsPolicyFEC  |
         |   +-----------------+
         |      |
         |      |  +----------------------------------+
         |      \--|  FilterList                        |
         |         |    TrafficType=IPv4              |
         |         |    SourceIPAddress=1.2.3.4       |
         |         |    DestinationIPAddress=2.3.4.6  |
         |         +----------------------------------+
         |
         |   +---------------------------+
         |-- |  qosPolicyTrfcProf        |
         |   |   qpPRRate=10Mb/s         |
         |   +---------------------------+
         |
         |   +---------------------------------+
         \-- |  mplsPolicyRouteSpec             |
             |    mpIngressIPAddress=1.2.3.5   |
             |    mpEgressIPAddress=2.3.4.5    |
             +---------------------------------+
```

Figure 5: Representation of the example policy rule

The policy object illustrated in Figure 5 is built from policy classes defined in CIM, QPIM and this document. The root of the policy object is a rule object that is associated with no conditions (i.e. provisioning case) and on LSP establishing action (mplsPolicyLSPAction). The LSP is specified by an instance of mplsPolicyLSP. The specification includes FEC specification (mplsPolicyFEC), a specification of the LSP traffic profile of the LSP (qosPolicyTrafficProfile) and the route specification (mplsPolicyRouteSpec).

6.2. Network wide bandwidth adjustment example

The following example demonstrates the use of our information model to define policies that are applied on a network-wide basis. In this example, we describe the implementation of a network-wide policy that varies bandwidth allocations based on time of day and type of traffic using the policy information model. Traffic Trunks in the network are assigned "roles" in accordance with the Olympic Services Model, i.e. they are marked as "gold", "silver" or "bronze" according to the QoS properties of the traffic that they carry. A network-wide policy controls the bandwidth allocation associated with these traffic trunks.

Specifically, the bandwidth for all gold TTs is reduced by 50% during nights and weekends. This could be potentially useful if the Service Level Agreements are such that a number of customers require gold service during business hours on weekdays, while they subscribe to lower service levels during non-business hours and weekends.

Policy rules are defined to describe the time-varying bandwidth allocations to the appropriate Traffic Trunks. Associated with each traffic trunk is a traffic profile that describes the resource requirements for this trunk. Also associated with each traffic trunk are one or more route specifications that specify possible ways of routing this traffic trunk through the network. Each such route specification has its hops specified.

For purposes of illustration, we assume the presence of a traffic engineering module that uses these specifications, along with information about the current nodes and links in the network, to compute appropriate constrained routes from these route specifications. In other words, not all route specifications will necessarily be used to create LSPs. Each traffic trunk is assigned to one or more LSPs according to its bandwidth requirements.

When the TT's bandwidth requirements are modified, the traffic engineering module attempts to resize the LSP(s) for this TT to accommodate the new bandwidth demand. If the existing LSPs are not adequate, the traffic engineering module performs computations so that new LSPs are established, in the manner described above.

Note that another scenario could be that the policy information model and the traffic provide no route specifications engineering module computes suitable LSPs based on traffic trunk attributes alone.

For this example, the policy-based system contains the following:

- A number of activated TTs (instance of mplsPolicyTrafficTrunk), each one assigned to one or more already established LSPs (instance of mplsPolicyLSP). For the assignment association an instance of mplsPolicyCurrentlyAssignedLSP association is used.

- Each TT is characterized as either "gold", "silver" or "bronze", according to the QoS requirements of the traffic that is being forwarded through it, using its Roles property to store this value.
- Each TT is associated (instance of mplsPolicyTT_TrafficProfile association) with one traffic profile (instance of qosPolicyPRTrfcProf).
- Each TT is associated (instance of mplsPolicyEligibleRouteSpec association) with zero or more eligible route specifications (instance of mplsPolicyRouteSpec).
- Each LSP is associated (instance of mplsPolicyLSPTrafficProfile association) with one traffic profile.

To implement the network-wide policy described above, the following two policy rules are required for the gold TTs:

```
Policy Rule 1: Roles property is set to "gold TT"
        IF "time is 8am" AND ("day is Monday" OR "day is Tuesday"
                        OR "day is Wednesday" OR "day is Thursday"
                        OR "day is Friday")
        THEN "Modify TT: Increase traffic profile bandwidth by 100%"

Policy Rule 2: Roles property is set to "gold TT"
        IF "time is 5pm" AND ("day is Monday" OR "day is Tuesday"
                        OR "day is Wednesday" OR "day is Thursday"
                        OR "day is Friday")
        THEN "Modify TT: Decrease traffic profile bandwidth by 50%"
```

The semantics of the first rule is that when the condition becomes true, it modifies the traffic profile of all the gold TTs, such that their bandwidth is increased by 100%.

In a similar fashion, the semantics of the second rule is that when the condition becomes true, it modifies the traffic profile of all the gold TTs such that their bandwidths get decreased by 50%. Note that the increase and decrease is the same absolute amount.

Note that the "Roles" property of those two policy rules is set to "gold TT" which indicates which TTs those rules are to be applied to. In this case those two rules are applied to all the TTs that have their Roles property also set to "gold TT".

When the traffic profiles of the gold TTs are modified, the traffic engineering module is activated, to verify whether the current mapping of TTs to LSPs is still valid. For each modified TT, the traffic engineering module might either increase the bandwidth of the currently assigned LSPs, or reroute it based on its attributes.

For modeling each of the above policy rules we use an instance of

PolicyRule (defined in [PCIM]), an instance of
PolicyTimePeriodCondition (also from [PCIM]) to represent the
condition part of the rule, and we have an instance of
mplsPolicyTTAction class to represent the action part of the rule,

with its mpTTmode property set to "TTModify". This action takes the current TTs in the context of the action and modifies the traffic profile of this TT by the given amount. Note that this poses some problem with the current PCIM; see the section on open issues for a description of this.

The Roles property is set to "gold TT" for the two rules (gold TT). In this way, the two rules will be applied to all the TTs that are considered gold and therefore have their Roles property set to "gold TT".

7. Security Considerations

The security considerations for this document are the same as those of [PCIM] and are not further addressed in this version of the draft.

8. Open Issues

The following open issues need to be resolved:

1) Control of Signaling protocol mechanisms: While the draft is independent of the signaling protocol used for label distribution, policy actions relating to the control of specific CR-LDP mechanisms are incorporated in this version. These mechanisms may be moved to a protocol-specific model in later versions.

2) The current draft contains traffic parameters specific to CR-LDP, as well as generic traffic parameters. Future versions may include additional parameters specific to other signaling protocols such as RSVP-TE. Further discussion is required to assess whether this is within the scope of the information model. Furthermore, this needs to be discussed with the PQIM authors.

3) As discussed in the example, we want to calculate a new value for the bandwidth property of the traffic trunk's traffic profile. This raises three questions: how do we refer to the traffic profile in the context of the action, how do we access a single property of the traffic profile, and how do we calculate a new value for the traffic profile based on its old value? We initially considered introducing specific actions to do this, but decided to wait until a PCIM extension (e.g., as proposed in [PCIM_EXT]) is discussed. Basically, the same problem arises if a policy wants to add new traffic to a TT or LSP (change the FEC by adding new filter entries etc.)

4) There may be a need to introduce a new class to model a Label Switched Router (LSR). This class would be derived from ComputerSystem in CIM.

5) mplsPolicyFEC and qosPolicyTrfcProf are derived from Policy.

Should they be derived from CIM's LogicalElement?

6) In a future version, we will replace all the references in the actions with associations.

7) Section 4.2.2 Traffic Trunks: "If a traffic trunk going in the reverse direction exists, it is associated with this traffic trunk via the mplsPolicyReverseDirTT association." What is the precise meaning of "in the reverse direction"? Possible meanings include reverse Ingress/Egress ProtocolEndpoints; reverse route; the reverse of the loosely specified route.

8) For the classes mplsPolicyLSP, mplsPolicyTrafficTrunc, and mplsPolicyProtocolEndpoint, we have defined a role property "Roles". First question, does PCIM support roles for objects not derived from the class System in CIM? Second, is there a naming convention to be used in order to meet PCIMs definition? If no, we will change the name into unique names in order to more easily map the model into different data representions such as LDAP.

9) In order to deal with DiffServ over MPLS, future work should take into account the mapping from LSPs to PHBs, and mapping of packets in LSPs to different PHBs (in case of E-LSPs [DS-MPLS]). Are there other means needed in this area?

9. References

[CIM]   Distributed Management Task Force, Inc., "Common Information Model (CIM) Schema, version 2.3, March 2000. The components of the CIM v2.3 schema are available via links on the following DMTF web page: http://www.dmtf.org/spec/cims.html

[PCIM] B. Moore, E. Ellesson, J. Strassner, "Policy Core Information Model -- Version 1 Specification", Internet Draft, <draft-ietf-policy-core-info-model-06.txt>, May, 2000.

[PQIM] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, "Policy Framework QoS Information Model", Internet draft, <draft-ietf-policy-qos-info-model-01.txt>, April 2000.

[CRLDP] B. Jamoussi, "Constraint-Based LSP Setup using LDP", Internet Draft, <draft-ietf-mpls-cr-ldp-p3.txt>, March 2000.

[RSVP-TE] Awduche, D.O., Berger, L., Gan, D., Li, T., Srinivasan, V., Swallow, G., "RSVP-TE: Extensions to RSVP for LSP Tunnels", Internet Draft draft-mpls-rsvp-lsp-tunnel-06.txt, July 2000.

[DS-MPLS] F. LeFaucher, L. Wu, B. Davie, S. Davari, P. Vaanenen, R. Krishnan, P. Cheval, J. Heinanen, "MPLS Support of Differentiated Services", work in progrss, draft-ietf-mpls-diff-ext-05.txt, June 2000.

[MPLS-ARCH] E.Rosen, A.Viswanathan, R.Callon, "Multiprotocol Label

          Switching Architecture", work in progress,
          draft-ietf-mpls-arch-06.txt, August 1999.

  [MPLS-FW] R.Callon, P.Doolan, N.Feldman, A.Freddette, G.Swallow, A.Viswanathan, "A Framework for MPLS", work in progress, draft-ietf-mpls-framework-05.txt, September 1999.

[REQPMPLS] S. Wright, S. Herzog, F. Reichmayer, R. Jaeger, "Requirements for Policy Enabled MPLS", work in progress, draft-wright-policy-mpls-00.txt, March 2000.

[REQMPLS_TE] Wright, S., Reichmeyer, F., Jaeger, R., Gibson, M., "Policy-Based Load-Balancing in Traffic-Engineered MPLS Networks", Internet Draft draft-wright-mpls-te-policy-00.txt, June 2000.

[MPLS-MIB] Srinivasan, C., Viswanathan, A., Nadeau, T.D., "MPLS Traffic Engineering Management Information Base Using SMIv2", Internet Draft, draft-ietf-mpls-te-mib-03.txt, March 2000.

[RFC2702] D. Awduche, J. Malcom, J. Agogbua, M. O'Dell, J. McManus, "Requirements for Traffic Engineering over MPLS", RFC 2702, September 1999.

[RFC2430] Li, T. and Y. Rekhter, "Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)", RFC 2430, October 1998.

[PCIM_EXT] M. Brunner, J. Quittek, "Policy Framework Core Info Model Extensions", Internet Draft, draft-brunner-policy-core-ext-00.txt, November 2000.

10. Authors' Addresses

Kazuhiko Isoyama, Makiko Yoshida
NEC Corporation
Development Laboratories
1131, Hinode, Abiko, Chiba, 270-1198, JAPAN
Phone: +81 471-85-6738
Fax: +81 471-85-6841
Email: [iso|myoshida]@ptl.abk.nec.co.jp

Marcus Brunner, Juergen Quittek
NEC Europe Ltd.
C&C Research Laboratories
Adenauerplatz 6
D-69115 Heidelberg, Germany
Phone: +49 (0)6221 905110
Fax: +49 (0)6221 9051155
Email: [brunner|quittek]@ccrle.nec.de

Ritu Chadha, George Mykoniatis, Alex Poylisher, Ravichander
Vaidyanathan
   Telcordia Technologies
   445 South Street

Morristown NJ 07960
Phone: +1-973-829-4869
Email: [chadha|mykoniat|sher|vravi]@research.telcordia.com

Andreas Kind
IBM Zurich Research Laboratory
Saumerstrasse 4
CH-8803 Ruschlikon
Switzerland
Phone: +41-1-724-8915
Fax +41-1-724-8578
Email: ank@zurich.ibm.com

Francis Reichmeyer
PfN, Inc.
26 Landsdowne St.
Cambridge, MA 02139
Phone: +1-617-529-3970
Email: franr@pfn.com

Full Copyright Statement