          Policy Core Lightweight Directory Access Protocol (LDAP) Schema

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   This document defines a mapping of the Policy Core Information Model
   to a form that can be implemented in a directory that uses
   Lightweight Directory Access Protocol (LDAP) as its access protocol.
   This model defines two hierarchies of object classes: structural
   classes representing information for representing and controlling
   policy data as specified in RFC 3060, and relationship classes that
   indicate how instances of the structural classes are related to each
   other. Classes are also added to the LDAP schema to improve the
   performance of a client's interactions with an LDAP server when the
   client is retrieving large amounts of policy-related information.
   These classes exist only to optimize LDAP retrievals: there are no
   classes in the information model that correspond to them.

Table of Contents

1. Introduction

   This document takes as its starting point the object-oriented
   information model for representing information for representing and
   controlling policy data as specified in [1]. Lightweight Directory
   Access Protocol (LDAP) [2] implementers, please note that the use of
   the term "policy" in this document does not refer to the use of the
   term "policy" as defined in X.501 [4]. Rather, the use of the term
   "policy" throughout this document is defined as follows:

Policy is defined as a set of rules to administer, manage, and
control access to network resources.

This work is currently under joint development in the IETF's Policy
Framework working group and in the Policy working group of the
Distributed Management Task Force (DMTF). This model defines two
hierarchies of object classes: structural classes representing policy
information and control of policies, and relationship classes that
indicate how instances of the structural classes are related to each
other. In general, both of these class hierarchies will need to be
mapped to a particular data store.

This document defines the mapping of these information model classes
to a directory that uses LDAP as its access protocol. Two types of
mappings are involved:

  - For the structural classes in the information model, the
     mapping is basically one-for-one: information model classes map
     to LDAP classes, information model properties map to LDAP
     attributes.

  - For the relationship classes in the information model,
     different mappings are possible. In this document, the Policy
     Core Information Model's (PCIM's) relationship classes and
     their properties are mapped in three ways: to LDAP auxiliary
     classes, to attributes representing distinguished name (DN)
     references, and to superior-subordinate relationships in the
     Directory Information Tree (DIT).

Implementations that use an LDAP directory as their policy repository
and want to implement policy information according to RFC 3060 [1]
SHALL use the LDAP schema defined in this document, or a schema that
subclasses from the schema defined in this document. The use of the
information model defined in reference [1] as the starting point
enables the inheritance and the relationship class hierarchies to be
extensible, such that other types of policy repositories, such as
relational databases, can also use this information.

This document fits into the overall framework for representing,
deploying, and managing policies being developed by the Policy
Framework Working Group.

The LDAP schema described in this document uses the prefix "pcim" to
identify its classes and attributes. It consists of ten very general
classes: pcimPolicy (an abstract class), three policy group classes
(pcimGroup, pcimGroupAuxClass, and pcimGroupInstance), three policy
rule classes (pcimRule, pcimRuleAuxClass, and pcimRuleInstance), and
three special auxiliary classes (pcimConditionAuxClass,

pcimTPCAuxClass, and pcimActionAuxClass). (Note that the
PolicyTimePeriodCondition auxiliary class defined in [1] would
normally have been named pcimTimePeriodConditionAuxClass, but this
name is too long for some directories. Therefore, we have
abbreviated this name to be pcimTPCAuxClass).

The mapping for the PCIM classes pcimGroup and pcimRule is designed
to be as flexible as possible. Three classes are defined for these
two PCIM classes. First, an abstract superclass is defined that
contains all required properties of each PCIM class. Then, both an
auxiliary class as well as a structural class are derived from the
abstract superclass. This provides maximum flexibility for the
developer.

The schema also contains two less general classes:
pcimConditionVendorAuxClass and pcimActionVendorAuxClass. To achieve
the mapping of the information model's relationships, the schema also
contains two auxiliary classes: pcimGroupContainmentAuxClass and
pcimRuleContainmentAuxClass. Capturing the distinction between
rule-specific and reusable policy conditions and policy actions
introduces seven other classes: pcimRuleConditionAssociation,
pcimRuleValidityAssociation, pcimRuleActionAssociation,
pcimPolicyInstance, and three policy repository classes
(pcimRepository, pcimRepositoryAuxClass, and pcimRepositoryInstance).
Finally, the schema includes two classes (pcimSubtreesPtrAuxClass and
pcimElementAuxClass) for optimizing LDAP retrievals. In all, the
schema contains 23 classes.

Within the context of this document, the term "PCLS" (Policy Core
LDAP Schema) is used to refer to the LDAP class definitions that this
document contains. The term "PCIM" refers to classes defined in [1].

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 [10].

2. The Policy Core Information Model

This document contains an LDAP schema representing the classes
defined in the companion document "Policy Core Information
Model -- Version 1 Specification" [1]. Other documents may
subsequently be produced, with mappings of this same PCIM to other
storage technologies. Since the detailed semantics of the PCIM
classes appear only in [1], that document is a prerequisite for
reading and understanding this document.

3. Inheritance Hierarchy for the PCLS

    The following diagram illustrates the class hierarchy for the LDAP
    Classes defined in this document:

```
     top
      |
      +--dlm1ManagedElement (abstract)
      |   |
      |   +--pcimPolicy (abstract)
      |   |   |
      |   |   +--pcimGroup (abstract)
      |   |   |   |
      |   |   |   +--pcimGroupAuxClass (auxiliary)
      |   |   |   |
      |   |   |   +--pcimGroupInstance (structural)
      |   |   |
      |   |   +--pcimRule (abstract)
      |   |   |   |
      |   |   |   +--pcimRuleAuxClass (auxiliary)
      |   |   |   |
      |   |   |   +--pcimRuleInstance (structural)
      |   |   |
      |   |   +--pcimRuleConditionAssociation (structural)
      |   |   |
      |   |   +--pcimRuleValidityAssociation (structural)
      |   |   |
      |   |   +--pcimRuleActionAssociation (structural)
      |   |   |
      |   |   +--pcimPolicyInstance (structural)
      |   |   |
      |   |   +--pcimElementAuxClass (auxiliary)
      |   |
      |   +--dlm1ManagedSystemElement (abstract)
      |       |
      |       +--dlm1LogicalElement (abstract)
      |           |
      |           +--dlm1System (abstract)
      |               |
      |               +--dlm1AdminDomain (abstract)
      |                   |
      |                   +--pcimRepository (abstract)
      |                       |
      |                       +--pcimRepositoryAuxClass (auxiliary)
```

```
        top
         |                          |
         |                          +--pcimRepositoryInstance
         |                             (structural)
         |
       +--pcimConditionAuxClass (auxiliary)
         |    |
         |    +---pcimTPCAuxClass (auxiliary)
         |    |
         |    +---pcimConditionVendorAuxClass (auxiliary)
         |
       +--pcimActionAuxClass (auxiliary)
         |    |
         |    +---pcimActionVendorAuxClass (auxiliary)
         |
       +--pcimSubtreesPtrAuxClass (auxiliary)
         |
       +--pcimGroupContainmentAuxClass (auxiliary)
         |
       +--pcimRuleContainmentAuxClass (auxiliary)
```

          Figure 1. LDAP Class Inheritance Hierarchy for the PCLS

4. General Discussion of Mapping the Information Model to LDAP

   The classes described in Section 5 below contain certain
   optimizations for a directory that uses LDAP as its access protocol.
   One example of this is the use of auxiliary classes to represent some
   of the associations defined in the information model. Other data
   stores might need to implement these associations differently. A
   second example is the introduction of classes specifically designed
   to optimize retrieval of large amounts of policy-related data from a
   directory. This section discusses some general topics related to the
   mapping from the information model to LDAP.

   The remainder of this section will discuss the following topics.
   Section 4.1 will discuss the strategy used in mapping the classes and
   associations defined in [1] to a form that can be represented in a
   directory that uses LDAP as its access protocol. Section 4.2
   discusses DIT content and structure rules, as well as name forms.
   Section 4.3 describes the strategy used in defining naming attributes
   for the schema described in Section 5 of this document. Section 4.4
   defines the strategy recommended for locating and retrieving
   PCIM-derived objects in the directory.

4.1. Summary of Class and Association Mappings

   Fifteen of the classes in the PCLS come directly from the nine
   corresponding classes in the information model. Note that names of
   classes begin with an upper case character in the information model
   (although for CIM in particular, case is not significant in class and
   property names), but with a lower case character in LDAP. This is
   because although LDAP doesn't care, X.500 doesn't allow class names
   to begin with an uppercase character. Note also that the prefix
   "pcim" is used to identify these LDAP classes.

```
       +------------------------------+------------------------------+
       | Information Model            | LDAP Class(es)               |
       +------------------------------+------------------------------+
       +------------------------------+------------------------------+
       | Policy                       | pcimPolicy                   |
       +------------------------------+------------------------------+
       | PolicyGroup                  | pcimGroup                    |
       |                              |    pcimGroupAuxClass         |
       |                              |    pcimGroupInstance         |
       +------------------------------+------------------------------+
       | PolicyRule                   | pcimRule                     |
       |                              |    pcimRuleAuxClass          |
       |                              |    pcimRuleInstance          |
       +------------------------------+------------------------------+
       | PolicyCondition              | pcimConditionAuxClass        |
       +------------------------------+------------------------------+
       | PolicyAction                 | pcimActionAuxClass           |
       +------------------------------+------------------------------+
       | VendorPolicyCondition        | pcimConditionVendorAuxClass  |
       +------------------------------+------------------------------+
       | VendorPolicyAction           | pcimActionVendorAuxClass     |
       +------------------------------+------------------------------+
       | PolicyTimePeriodCondition    | pcimTPCAuxClass              |
       +------------------------------+------------------------------+
       | PolicyRepository             | pcimRepository               |
       |                              |    pcimRepositoryAuxClass     |
       |                              |    pcimRepositoryInstance     |
       +------------------------------+------------------------------+
```

            Figure 2. Mapping of Information Model Classes to LDAP

   The associations in the information model map to attributes that
   reference DNs (Distinguished Names) or to Directory Information Tree
   (DIT) containment (i.e., superior-subordinate relationships) in LDAP.
   Two of the attributes that reference DNs appear in auxiliary classes,
   which allow each of them to represent several relationships from the
   information model.

```
+----------------------------------+----------------------------------+
| Information Model Association     | LDAP Attribute / Class           |
+----------------------------------+----------------------------------+

+----------------------------------+----------------------------------+
| PolicyGroupInPolicyGroup         | pcimGroupsAuxContainedSet in     |
|                                  |   pcimGroupContainmentAuxClass    |
+----------------------------------+----------------------------------+
| PolicyRuleInPolicyGroup          | pcimRulesAuxContainedSet in      |
|                                  |   pcimRuleContainmentAuxClass    |
+----------------------------------+----------------------------------+
| PolicyConditionInPolicyRule      | DIT containment or               |
|                                  | pcimRuleConditionList in         |
|                                  |   pcimRule or                    |
|                                  | pcimConditionDN in               |
|                                  |   pcimRuleConditionAssociation   |
+----------------------------------+----------------------------------+
| PolicyActionInPolicyRule         | DIT containment or               |
|                                  | pcimRuleActionList in            |
|                                  |   pcimRule or                    |
|                                  | pcimActionDN in                  |
|                                  |   pcimRuleActionAssociation      |
+----------------------------------+----------------------------------+
| PolicyRuleValidityPeriod         | pcimRuleValidityPeriodList       |
|                                  |   in pcimRule or (if reusable)   |
|                                  |   referenced through the         |
|                                  | pcimTimePeriodConditionDN in     |
|                                  |   pcimRuleValidityAssociation    |
+----------------------------------+----------------------------------+
| PolicyConditionInPolicyRepository| DIT containment                  |
+----------------------------------+----------------------------------+
| PolicyActionInPolicyRepository   | DIT containment                  |
+----------------------------------+----------------------------------+
| PolicyRepositoryInPolicyRepository| DIT containment                 |
+----------------------------------+----------------------------------+
```

         Figure 3. Mapping of Information Model Associations to LDAP

   Of the remaining classes in the PCLS, two (pcimElementAuxClass and
   pcimSubtreesPtrAuxClass) are included to make navigation through the
   DIT and retrieval of the entries found there more efficient. This
   topic is discussed below in Section 4.5.

   The remaining four classes in the PCLS, pcimRuleConditionAssociation,
   pcimRuleValidityAssociation, pcimRuleActionAssociation, and
   pcimPolicyInstance, are all involved with the representation of
   policy conditions and policy actions in an LDAP directory. This
   topic is discussed below in Section 4.4.

4.2. Usage of DIT Content and Structure Rules and Name Forms

   There are three powerful tools that can be used to help define
   schemata. The first, DIT content rules, is a way of defining the
   content of an entry for a structural object class. It can be used to
   specify the following characteristics of the entry:

       - additional mandatory attributes that the entries are required
         to contain
       - additional optional attributes the entries are allowed to
         contain
       - the set of additional auxiliary object classes that these
         entries are allowed to be members of
       - any optional attributes from the structural and auxiliary
         object class definitions that the entries are required to
         preclude

   DIT content rules are NOT mandatory for any structural object class.

   A DIT structure rule, together with a name form, controls the
   placement and naming of an entry within the scope of a subschema.
   Name forms define which attribute type(s) are required and are
   allowed to be used in forming the Relative Distinguished Names (RDNs)
   of entries. DIT structure rules specify which entries are allowed to
   be superior to other entries, and hence control the way that RDNs are
   added together to make DNs.

   A name form specifies the following:

       - the structural object class of the entries named by this name
         form
       - attributes that are required to be used in forming the RDNs of
         these entries
       - attributes that are allowed to be used in forming the RDNs of
         these entries
       - an object identifier to uniquely identify this name form

   Note that name forms can only be specified for structural object
   classes. However, every entry in the DIT must have a name form
   controlling it.

   Unfortunately, current LDAP servers vary quite a lot in their support
   of these features. There are also three crucial implementation
   points that must be followed. First, X.500 use of structure rules
   requires that a structural object class with no superior structure
   rule be a subschema administrative point. This is exactly NOT what
   we want for policy information. Second, when an auxiliary class is
   subclassed, if a content rule exists for the structural class that

the auxiliary class refers to, then that content rule needs to be
augmented. Finally, most LDAP servers unfortunately do not support
inheritance of structure and content rules.

Given these concerns, DIT structure and content rules have been
removed from the PCLS. This is because, if included, they would be
normative references and would require OIDs. However, we don't want
to lose the insight gained in building the structure and content
rules of the previous version of the schema. Therefore, we describe
where such rules could be used in this schema, what they would
control, and what their effect would be.

4.3. Naming Attributes in the PCLS

Instances in a directory are identified by distinguished names (DNs),
which provide the same type of hierarchical organization that a file
system provides in a computer system. A distinguished name is a
sequence of RDNs. An RDN provides a unique identifier for an
instance within the context of its immediate superior, in the same
way that a filename provides a unique identifier for a file within
the context of the folder in which it resides.

To preserve maximum naming flexibility for policy administrators,
three optional (i.e., "MAY") naming attributes have been defined.
They are:

    - Each of the structural classes defined in this schema has its
       own unique ("MAY") naming attribute. Since the naming
       attributes are different, a policy administrator can, by using
       these attributes, guarantee that there will be no name
       collisions between instances of different classes, even if the
       same value is assigned to the instances' respective naming
       attributes.

    - The LDAP attribute cn (corresponding to X.500's commonName) is
       included as a MAY attribute in the abstract class pcimPolicy,
       and thus by inheritance in all of its subclasses. In X.500,
       commonName typically functions as an RDN attribute, for naming
       instances of many classes (e.g., X.500's person class).

    - A special attribute is provided for implementations that expect
       to map between native CIM and LDAP representations of policy
       information. This attribute, called orderedCimKeys, is defined
       in the class dlm1ManagedElement [6]. The value of this
       attribute is derived algorithmically from values that are
       already present in a CIM policy instance. The normative
       reference for this algorithm is contained in [6]. See the
       appendix of this document for a description of the algorithm.

Since any of these naming attributes MAY be used for naming an
instance of a PCLS class, implementations MUST be able to accommodate
instances named in any of these ways.

Note that it is recommended that two or more of these attributes
SHOULD NOT be used together to form a multi-part RDN, since support
for multi-part RDNs is limited among existing directory
implementations.

4.4. Rule-Specific and Reusable Conditions and Actions

The PCIM [1] distinguishes between two types of policy conditions and
policy actions: those associated with a single policy rule, and
those that are reusable, in the sense that they may be associated
with more than one policy rule. While there is no inherent
functional difference between a rule-specific condition or action and
a reusable one, there is both a usage, as well as, an implementation
difference between them.

Defining a condition or action as reusable vs. rule-specific reflects
a conscious decision on the part of the administrator in defining how
they are used. In addition, there are variations that reflect
implementing rule-specific vs. reusable policy conditions and actions
and how they are treated in a policy repository. The major
implementation differences between a rule-specific and a reusable
condition or action are delineated below:

1. It is natural for a rule-specific condition or action to be
    removed from the policy repository at the same time the rule is.
    It is just the opposite for reusable conditions and actions.
    This is because the condition or action is conceptually attached
    to the rule in the rule-specific case, whereas it is referenced
    (e.g., pointed at) in the reusable case. The persistence of a
    pcimRepository instance is independent of the persistence of a
    pcimRule instance.
2. Access permissions for a rule-specific condition or action are
    usually identical to those for the rule itself. On the other
    hand, access permissions of reusable conditions and actions must
    be expressible without reference to a policy rule.
3. Rule-specific conditions and actions require fewer accesses,
    because the conditions and actions are "attached" to the rule.
    In contrast, reusable conditions and actions require more
    accesses, because each condition or action that is reusable
    requires a separate access.
4. Rule-specific conditions and actions are designed for use by a
    single rule. As the number of rules that use the same
    rule-specific condition increase, subtle problems are created
    (the most obvious being how to keep the rule-specific conditions

and actions updated to reflect the same value). Reusable
conditions and actions lend themselves for use by multiple
independent rules.

5. Reusable conditions and actions offer an optimization when
    multiple rules are using the same condition or action. This is
    because the reusable condition or action only needs be updated
    once, and by virtue of DN reference, the policy rules will be
    automatically updated.

The preceding paragraph does not contain an exhaustive list of the
ways in which reusable and rule-specific conditions should be treated
differently. Its purpose is merely to justify making a semantic
distinction between rule-specific and reusable, and then reflecting
this distinction in the policy repository itself.

When the policy repository is realized in an LDAP-accessible
directory, the distinction between rule-specific and reusable
conditions and actions is realized via placement of auxiliary classes
and via DIT containment. Figure 4 illustrates a policy rule Rule1
with one rule-specific condition CA and one rule-specific action AB.

```
                   +-----+
                   |Rule1|
                   |     |
         +-----|-    -|-----+
         |     +-----+       |
         |       * *         |
         |       * *         |
         |     **** ****     |
         |      *       *    |
         v      *       *    v
     +--------+   +--------+
     | CA+ca  |   | AB+ab  |
     +--------+   +--------+



               +-----------------------------+
               |LEGEND:                      |
               |  ***** DIT containment      |
               |    +    auxiliary attachment |
               |  ----> DN reference         |
               +-----------------------------+
```
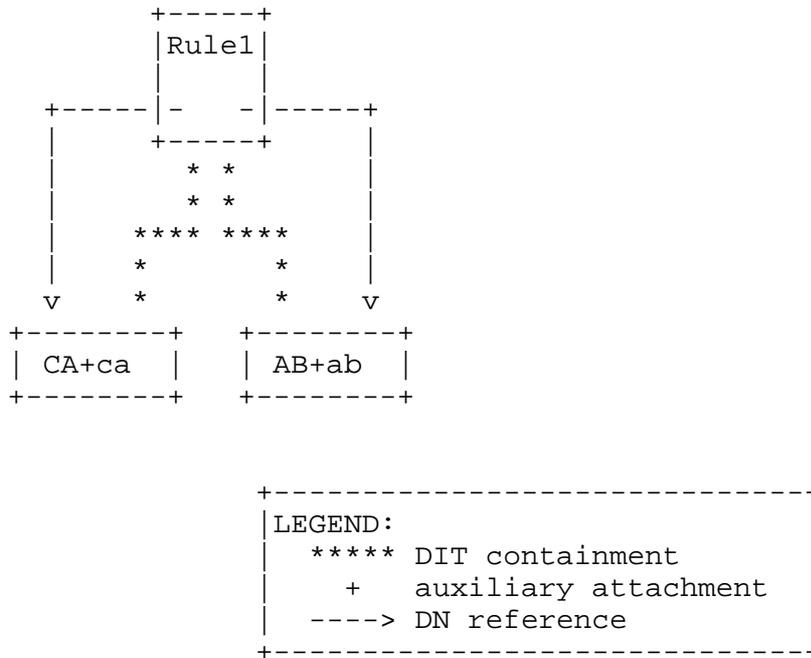
          Figure 4 Rule-Specific Policy Conditions and Actions

Because the condition and action are specific to Rule1, the auxiliary
classes ca and ab that represent them are attached, respectively, to
the structural classes CA and AB. These structural classes represent
not the condition ca and action ab themselves, but rather the
associations between Rule1 and ca, and between Rule1 and ab.

As Figure 4 illustrates, Rule1 contains DN references to the
structural classes CA and AB that appear below it in the DIT. At
first glance it might appear that these DN references are
unnecessary, since a subtree search below Rule1 would find all of the
structural classes representing the associations between Rule1 and
its conditions and actions. Relying only on a subtree search,
though, runs the risk of missing conditions or actions that should
have appeared in the subtree, but for some reason did not, or of
finding conditions or actions that were inadvertently placed in the
subtree, or that should have been removed from the subtree, but for
some reason were not. Implementation experience has suggested that
many (but not all) of these risks are eliminated.

However, it must be noted that this comes at a price. The use of DN
references, as shown in Figure 4 above, thwarts inheritance of access
control information as well as existence dependency information. It
also is subject to referential integrity considerations. Therefore,
it is being included as an option for the designer.

Figure 5 illustrates a second way of representing rule-specific
conditions and actions in an LDAP-accessible directory: attachment of
the auxiliary classes directly to the instance representing the
policy rule. When all of the conditions and actions are attached to
a policy rule in this way, the rule is termed a "simple" policy rule.
When conditions and actions are not attached directly to a policy
rule, the rule is termed a "complex" policy rule.

```
                    +-----------+
                    |Rule1+ca+ab|
                    |           |
                    +-----------+

                       +------------------------------+
                       |LEGEND:                       |
                       |    +   auxiliary attachment  |
                       +------------------------------+
```
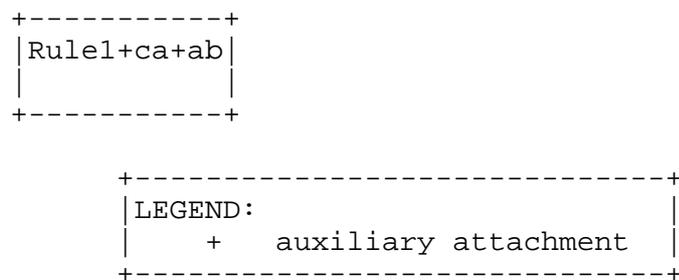
                 Figure 5. A Simple Policy Rule

The simple/complex distinction for a policy rule is not all or nothing. A policy rule may have its conditions attached to itself and its actions attached to other entries, or it may have its actions attached to itself and its conditions attached to other entries. However, it SHALL NOT have either its conditions or its actions attached both to itself and to other entries, with one exception: a policy rule may reference its validity periods with the pcimRuleValidityPeriodList attribute, but have its other conditions attached to itself.

The tradeoffs between simple and complex policy rules are between the efficiency of simple rules and the flexibility and greater potential for reuse of complex rules. With a simple policy rule, the semantic options are limited:

-   All conditions are ANDed together. This combination can be
    represented in two ways in the Disjunctive Normal Form (DNF)/
    Conjunctive Normal Form (CNF) (please see [1] for definitions of
    these terms) expressions characteristic of policy conditions: as
    a DNF expression with a single AND group, or as a CNF expression
    with multiple single-condition OR groups. The first of these is
    arbitrarily chosen as the representation for the ANDed conditions
    in a simple policy rule.

-   If multiple actions are included, no order can be specified for
    them.

If a policy administrator needs to combine conditions in some other way, or if there is a set of actions that must be ordered, then the only option is to use a complex policy rule.

Finally, Figure 6 illustrates the same policy rule Rule1, but this time its condition and action are reusable. The association classes CA and AB are still present, and they are still DIT contained under Rule1. But rather than having the auxiliary classes ca and ab attached directly to the association classes CA and AB, each now contains DN references to other entries to which these auxiliary classes are attached. These other entries, CIA and AIB, are DIT contained under RepositoryX, which is an instance of the class pcimRepository. Because they are named under an instance of pcimRepository, ca and ab are clearly identified as reusable.

```
                +------+              +-------------+
                |Rule1|              | RepositoryX |
            +-|-    -|--+            |             |
            |  +-----+  |            +-------------+
            |    *  *   |                *         *
            |    *  *   |                *         *
            |  *** ****  |               *         *
            |   *      *  v              *         *
            |   *    +---+               *         *
            |   *    |AB |           +------+      *
            v  *     |   -|-------->|AIB+ab|      *
           +---+     +---+           +------+      *
           |CA |                        +------+
           |   -|----------------------->|CIA+ca|
           +---+                        +------+


              +----------------------------+
              |LEGEND:                     |
              |   ***** DIT containment    |
              |     +   auxiliary attachment|
              |   ----> DN reference        |
              +----------------------------+
```
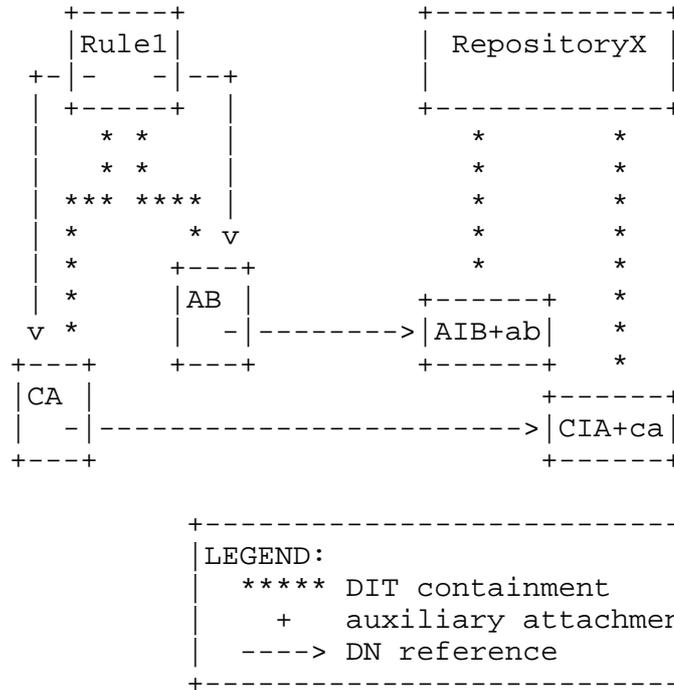
           Figure 6. Reusable Policy Conditions and Actions

   The classes pcimConditionAuxClass and pcimActionAuxClass do not
   themselves represent actual conditions and actions: these are
   introduced in their subclasses. What pcimConditionAuxClass and
   pcimActionAuxClass do introduce are the semantics of being a policy
   condition or a policy action. These are the semantics that all the
   subclasses of pcimConditionAuxClass and pcimActionAuxClass inherit.
   Among these semantics are those of representing either a
   rule-specific or a reusable policy condition or policy action.

   In order to preserve the ability to represent a rule-specific or a
   reusable condition or action, as well as a simple policy rule, all
   the subclasses of pcimConditionAuxClass and pcimActionAuxClass MUST
   also be auxiliary classes.

4.5. Location and Retrieval of Policy Objects in the Directory

   When a Policy Decision Point (PDP) goes to an LDAP directory to
   retrieve the policy object instances relevant to the Policy
   Enforcement Points (PEPs) it serves, it is faced with two related
   problems:

   -    How does it locate and retrieve the directory entries that apply
        to its PEPs? These entries may include instances of the PCLS
        classes, instances of domain-specific subclasses of these
        classes, and instances of other classes modeling such resources
        as user groups, interfaces, and address ranges.

   -    How does it retrieve the directory entries it needs in an
        efficient manner, so that retrieval of policy information from
        the directory does not become a roadblock to scalability? There
        are two facets to this efficiency: retrieving only the relevant
        directory entries, and retrieving these entries using as few LDAP
        calls as possible.

   The placement of objects in the Directory Information Tree (DIT)
   involves considerations other than how the policy-related objects
   will be retrieved by a PDP. Consequently, all that the PCLS can do
   is to provide a "toolkit" of classes to assist the policy
   administrator as the DIT is being designed and built. A PDP SHOULD
   be able to take advantage of any tools that the policy administrator
   is able to build into the DIT, but it MUST be able to use a less
   efficient means of retrieval if that is all it has available to it.

   The basic idea behind the LDAP optimization classes is a simple one:
   make it possible for a PDP to retrieve all the policy-related objects
   it needs, and only those objects, using as few LDAP calls as
   possible. An important assumption underlying this approach is that
   the policy administrator has sufficient control over the underlying
   DIT structure to define subtrees for storing policy information. If
   the policy administrator does not have this level of control over DIT
   structure, a PDP can still retrieve the policy-related objects it
   needs individually. But it will require more LDAP access operations
   to do the retrieval in this way. Figure 7 illustrates how LDAP
   optimization is accomplished.

```
                        +-----+
        --------------->|  A  |
     DN reference to    |     |     DN references to subtrees    +---+
     starting object    +-----+     +------------------------->| C |
                        |  o--+----+            +---+            +---+
                        |  o--+----------->| B |            /     \
                        +-----+            +---+           /       \
                       /       \          /     \         /  ...    \
                      /         \        /       \       /           \
                     /           \      /  ...    \
```
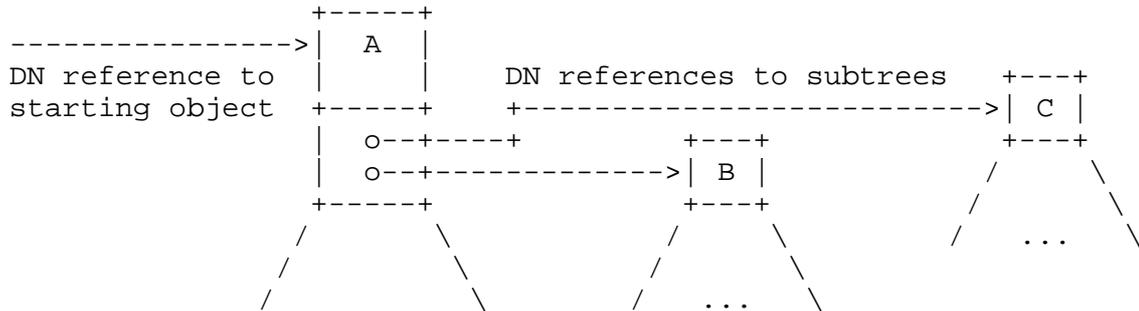
           Figure 7. Using the pcimSubtreesPtrAuxClass to Locate Policies

     The PDP is configured initially with a DN reference to some entry in
     the DIT. The structural class of this entry is not important; the
     PDP is interested only in the pcimSubtreesPtrAuxClass attached to it.
     This auxiliary class contains a multi-valued attribute with DN
     references to objects that anchor subtrees containing policy-related
     objects of interest to the PDP. Since pcimSubtreesPtrAuxClass is an
     auxiliary class, it can be attached to an entry that the PDP would
     need to access anyway - perhaps an entry containing initial
     configuration settings for the PDP, or for a PEP that uses the PDP.

     Once it has retrieved the DN references, the PDP will direct to each
     of the objects identified by them an LDAP request that all entries in
     its subtree be evaluated against the selection criteria specified in
     the request. The LDAP-enabled directory then returns all entries in
     that subtree that satisfy the specified criteria.

     The selection criteria always specify that object class="pcimPolicy".
     Since all classes representing policy rules, policy conditions, and
     policy actions, both in the PCLS and in any domain-specific schema
     derived from it, are subclasses of the abstract class policy, this
     criterion evaluates to TRUE for all instances of these classes. To
     accommodate special cases where a PDP needs to retrieve objects that
     are not inherently policy-related (for example, an IP address range
     object referenced by a subclass of pcimActionAuxClass representing
     the DHCP action "assign from this address range"), the auxiliary
     class pcimElementAuxClass can be used to "tag" an entry, so that it
     will be found by the selection criterion "object class=pcimPolicy".

     The approach described in the preceding paragraph will not work for
     certain directory implementations, because these implementations do
     not support matching of auxiliary classes in the objectClass
     attribute. For environments where these implementations are expected
     to be present, the "tagging" of entries as relevant to policy can be

accomplished by inserting the special value "POLICY" into the list of
values contained in the pcimKeywords attribute (provided by the
pcimPolicy class).

If a PDP needs only a subset of the policy-related objects in the
indicated subtrees, then it can be configured with additional
selection criteria based on the pcimKeywords attribute defined in the
pcimPolicy class. This attribute supports both standardized and
administrator- defined values. For example, a PDP could be
configured to request only those policy-related objects containing
the keywords "DHCP" and "Eastern US".

To optimize what is expected to be a typical case, the initial
request from the client includes not only the object to which its
"seed" DN references, but also the subtree contained under this
object. The filter for searching this subtree is whatever the client
is going to use later to search the other subtrees: object
class="pcimPolicy" or the presence of the keyword "POLICY", and/or
presence of a more specific value of pcimKeywords (e.g., "QoS Edge
Policy").

Returning to the example in Figure 7, we see that in the best case, a
PDP can get all the policy-related objects it needs, and only those
objects, with exactly three LDAP requests: one to its starting
object A to get the references to B and C, as well as the
policy-related objects it needs from the subtree under A, and then
one each to B and C to get all the policy-related objects that pass
the selection criteria with which it was configured. Once it has
retrieved all of these objects, the PDP can then traverse their
various DN references locally to understand the semantic
relationships among them. The PDP should also be prepared to find a
reference to another subtree attached to any of the objects it
retrieves, and to follow this reference first, before it follows any
of the semantically significant references it has received. This
recursion permits a structured approach to identifying related
policies. In Figure 7, for example, if the subtree under B includes
departmental policies and the one under C includes divisional
policies, then there might be a reference from the subtree under C to
an object D that roots the subtree of corporate-level policies.

A PDP SHOULD understand the pcimSubtreesPtrAuxClass class, SHOULD be
capable of retrieving and processing the entries in the subtrees it
references, and SHOULD be capable of doing all of this recursively.
The same requirements apply to any other entity needing to retrieve
policy information from the directory. Thus, a Policy Management
Tool that retrieves policy entries from the directory in order to
perform validation and conflict detection SHOULD also understand and
be capable of using the pcimSubtreesPtrAuxClass. All of these

requirements are "SHOULD"s rather than "MUST"s because an LDAP client
that doesn't implement them can still access and retrieve the
directory entries it needs. The process of doing so will just be
less efficient than it would have been if the client had implemented
these optimizations.

When it is serving as a tool for creating policy entries in the
directory, a Policy Management Tool SHOULD support creation of
pcimSubtreesPtrAuxClass entries and their references to object
instances.

4.5.1. Aliases and Other DIT-Optimization Techniques

Additional flexibility in DIT structure is available to the policy
administrator via LDAP aliasing and other techniques. Previous
versions of this document have used aliases. However, because
aliases are experimental, the use of aliases has been removed from
this version of this document. This is because the IETF has yet to
produce a specification on how aliases are represented in the
directory or how server implementations are to process aliases.

5. Class Definitions

The semantics for the policy information classes that are to be
mapped directly from the information model to an LDAP representation
are detailed in [1]. Consequently, all that this document presents
for these classes is the specification for how to do the mapping from
the information model (which is independent of repository type and
access protocol) to a form that can be accessed using LDAP. Remember
that some new classes needed to be created (that were not part of
[1]) to implement the LDAP mapping. These new LDAP-only classes are
fully documented in this document.

The formal language for specifying the classes, attributes, and DIT
structure and content rules is that defined in reference [3]. If
your implementation does not support auxiliary class inheritance, you
will have to list auxiliary classes in content rules explicitly or
define them in another (implementation-specific) way.

The following notes apply to this section in its entirety.

Note 1: in the following definitions, the class and attribute
definitions follow RFC 2252 [3] but they are line-wrapped to enhance
human readability.

Note 2: where applicable, the possibilities for specifying DIT
structure and content rules are noted. However, care must be taken
in specifying DIT structure rules. This is because X.501 [4] states

that an entry may only exist in the DIT as a subordinate to another
superior entry (the superior) if a DIT structure rule exists in the
governing subschema which:

1) indicates a name form for the structural object class of the
    subordinate entry, and
2) either includes the entry's superior structure rule as a possible
    superior structure rule, or
3) does not specify a superior structure rule.

If this last case (3) applies, then the entry is defined to be a
subschema administrative point. This is not what is desired.
Therefore, care must be taken in defining structure rules, and in
particular, they must be locally augmented.

Note 3: Wherever possible, both an equality and a substring matching
rule are defined for a particular attribute (as well as an ordering
match rule to enable sorting of matching results). This provides two
different choices for the developer for maximum flexibility.

For example, consider the pcimRoles attribute (section 5.3). Suppose
that a PEP has reported that it is interested in pcimRules for three
roles R1, R2, and R3. If the goal is to minimize queries, then the
PDP can supply three substring filters containing the three role
names.

These queries will return all of the pcimRules that apply to the PEP,
but they may also get some that do not apply (e.g., ones that contain
one of the roles R1, R2, or R3 and one or more other roles present in
a role-combination [1]).

Another strategy would be for the PDP to use only equality filters.
This approach eliminates the extraneous replies, but it requires the
PDP to explicitly build the desired role-combinations itself. It
also requires extra queries. Note that this approach is practical
only because the role names in a role combination are required to
appear in alphabetical order.

Note 4: in the following definitions, note that all LDAP matching
rules are defined in [3] and in [9]. The corresponding X.500
matching rules are defined in [8].

Note 5: some of the following attribute definitions specify
additional constraints on various data types (e.g., this integer has
values that are valid from 1..10). Text has been added to instruct
servers and applications what to do if a value outside of this range

   is encountered. In all cases, if a constraint is violated, then the
   policy rule SHOULD be treated as being disabled, meaning that
   execution of the policy rule SHOULD be stopped.

5.1. The Abstract Class pcimPolicy

   The abstract class pcimPolicy is a direct mapping of the abstract
   class Policy from the PCIM. The class value "pcimPolicy" is also
   used as the mechanism for identifying policy-related instances in the
   Directory Information Tree. An instance of any class may be "tagged"
   with this class value by attaching to it the auxiliary class
   pcimElementAuxClass. Since pcimPolicy is derived from the class
   dlm1ManagedElement defined in reference [6], this specification has a
   normative dependency on that element of reference [6].

   The class definition is as follows:

        (1.3.6.1.1.6.1.1 NAME 'pcimPolicy'
          DESC 'An abstract class that is the base class for all classes
                that describe policy-related instances.'
          SUP dlm1ManagedElement
          ABSTRACT
          MAY (cn $ dlmCaption $ dlmDescription $ orderedCimKeys $
                pcimKeywords)
)

   The attribute cn is defined in RFC 2256 [7]. The dlmCaption,
   dlmDescription, and orderedCimKeys attributes are defined in [6].

   The pcimKeywords attribute is a multi-valued attribute that contains
   a set of keywords to assist directory clients in locating the policy
   objects identified by these keywords. It is defined as follows:

        (1.3.6.1.1.6.2.3 NAME 'pcimKeywords'
              DESC 'A set of keywords to assist directory clients in
                    locating the policy objects applicable to them.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
)

5.2. The Three Policy Group Classes

   PCIM [1] defines the PolicyGroup class to serve as a generalized
   aggregation mechanism, enabling PolicyRules and/or PolicyGroups to be
   aggregated together. PCLS maps this class into three LDAP classes,
   called pcimGroup, pcimGroupAuxClass, and pcimGroupInstance. This is
   done in order to provide maximum flexibility for the DIT designer.

   The class definitions for the three policy group classes are listed
   below. These class definitions do not include attributes to realize
   the PolicyRuleInPolicyGroup and PolicyGroupInPolicyGroup associations
   from the PCIM. This is because a pcimGroup object refers to
   instances of pcimGroup and pcimRule via, respectively, the attribute
   pcimGroupsAuxContainedSet in the pcimGroupContainmentAuxClass object
   class and the attribute pcimRulesAuxContainedSet in the
   pcimRuleContainmentAuxClass object class.

   To maximize flexibility, the pcimGroup class is defined as abstract.
   The subclass pcimGroupAuxClass provides for auxiliary attachment to
   another entry, while the structural subclass pcimGroupInstance is
   available to represent a policy group as a standalone entry.

   The class definitions are as follows. First, the definition of the
   abstract class pcimGroup:

        (1.3.6.1.1.6.1.2 NAME 'pcimGroup'
              DESC 'A container for a set of related pcimRules and/or
                    a set of related pcimGroups.'
              SUP pcimPolicy
              ABSTRACT
              MAY (pcimGroupName)
)

   The one attribute of pcimGroup is pcimGroupName. This attribute is
   used to define a user-friendly name of this policy group, and may be
   used as a naming attribute if desired. It is defined as follows:

        (1.3.6.1.1.6.2.4 NAME 'pcimGroupName'
              DESC 'The user-friendly name of this policy group.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
)

The two subclasses of pcimGroup are defined as follows. The class
pcimGroupAuxClass is an auxiliary class that can be used to collect a
set of related pcimRule and/or pcimGroup classes. It is defined as
follows:

```
( 1.3.6.1.1.6.1.3 NAME 'pcimGroupAuxClass'
      DESC 'An auxiliary class that collects a set of related
            pcimRule and/or pcimGroup entries.'
      SUP pcimGroup
      AUXILIARY
)
```

The class pcimGroupInstance is a structural class that can be used to
collect a set of related pcimRule and/or pcimGroup classes. It is
defined as follows:

```
( 1.3.6.1.1.6.1.4 NAME 'pcimGroupInstance'
      DESC 'A structural class that collects a set of related
            pcimRule and/or pcimGroup entries.'
      SUP pcimGroup
      STRUCTURAL
)
```

A DIT content rule could be written to enable an instance of
pcimGroupInstance to have attached to it either references to one or
more policy groups (using pcimGroupContainmentAuxClass) or references
to one or more policy rules (using pcimRuleContainmentAuxClass).
This would be used to formalize the semantics of the PolicyGroup
class [1]. Since these semantics do not include specifying any
properties of the PolicyGroup class, the content rule would not need
to specify any attributes.

Similarly, three separate DIT structure rules could be written, each
of which would refer to a specific name form that identified one of
the three possible naming attributes (i.e., pcimGroupName, cn, and
orderedCIMKeys) for the pcimGroup object class. This structure rule
SHOULD include a superiorStructureRule (see Note 2 at the beginning
of section 5). The three name forms referenced by the three
structure rules would each define one of the three naming attributes.

5.3. The Three Policy Rule Classes

The information model defines a PolicyRule class to represent the "If
Condition then Action" semantics associated with processing policy
information. For maximum flexibility, the PCLS maps this class into
three LDAP classes.

To maximize flexibility, the pcimRule class is defined as abstract.
The subclass pcimRuleAuxClass provides for auxiliary attachment to
another entry, while the structural subclass pcimRuleInstance is
available to represent a policy rule as a standalone entry.

The conditions and actions associated with a policy rule are modeled,
respectively, with auxiliary subclasses of the auxiliary classes
pcimConditionAuxClass and pcimActionAuxClass. Each of these
auxiliary subclasses is attached to an instance of one of three
structural classes. A subclass of pcimConditionAuxClass is attached
to an instance of pcimRuleInstance, to an instance of
pcimRuleConditionAssociation, or to an instance of
pcimPolicyInstance. Similarly, a subclass of pcimActionAuxClass is
attached to an instance of pcimRuleInstance, to an instance of
pcimRuleActionAssociation, or to an instance of pcimPolicyInstance.

The pcimRuleValidityPeriodList attribute (defined below) realizes the
PolicyRuleValidityPeriod association defined in the PCIM. Since this
association has no additional properties besides those that tie the
association to its associated objects, this association can be
realized by simply using an attribute. Thus, the
pcimRuleValidityPeriodList attribute is simply a multi-valued
attribute that provides an unordered set of DN references to one or
more instances of the pcimTPCAuxClass, indicating when the policy
rule is scheduled to be active and when it is scheduled to be
inactive. A policy rule is scheduled to be active if it is active
according to AT LEAST ONE of the pcimTPCAuxClass instances referenced
by this attribute.

The PolicyConditionInPolicyRule and PolicyActionInPolicyRule
associations, however, do have additional attributes. The
association PolicyActionInPolicyRule defines an integer attribute to
sequence the actions, and the association PolicyConditionInPolicyRule
has both an integer attribute to group the condition terms as well as
a Boolean property to specify whether a condition is to be negated.

In the PCLS, these additional association attributes are represented
as attributes of two classes introduced specifically to model these
associations. These classes are the pcimRuleConditionAssociation
class and the pcimRuleActionAssociation class, which are defined in
Sections 5.4 and 5.5, respectively. Thus, they do not appear as
attributes of the class pcimRule. Instead, the pcimRuleConditionList
and pcimRuleActionList attributes can be used to reference these
classes.

The class definitions for the three pcimRule classes are as follows.

The abstract class pcimRule is a base class for representing the "If
Condition then Action" semantics associated with a policy rule. It
is defined as follows:

```
( 1.3.6.1.1.6.1.5 NAME 'pcimRule'
      DESC 'The base class for representing the "If Condition
            then Action" semantics associated with a policy rule.'
      SUP pcimPolicy
      ABSTRACT
      MAY ( pcimRuleName $ pcimRuleEnabled $
            pcimRuleConditionListType $ pcimRuleConditionList $
            pcimRuleActionList $ pcimRuleValidityPeriodList $
            pcimRuleUsage $ pcimRulePriority $
            pcimRuleMandatory $ pcimRuleSequencedActions $
            pcimRoles )
)
```

The PCIM [1] defines seven properties for the PolicyRule class. The
PCLS defines eleven attributes for the pcimRule class, which is the
LDAP equivalent of the PolicyRule class. Of these eleven attributes,
seven are mapped directly from corresponding properties in PCIM's
PolicyRule class. The remaining four attributes are a class-specific
optional naming attribute, and three attributes used to realize the
three associations that the pcimRule class participates in.

The pcimRuleName attribute is used as a user-friendly name of this
policy rule, and can also serve as the class-specific optional naming
attribute. It is defined as follows:

```
( 1.3.6.1.1.6.2.5 NAME 'pcimRuleName'
      DESC 'The user-friendly name of this policy rule.'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
      SINGLE-VALUE
)
```

The pcimRuleEnabled attribute is an integer enumeration indicating
whether a policy rule is administratively enabled (value=1),
administratively disabled (value=2), or enabled for debug (value=3).
It is defined as follows:

```
( 1.3.6.1.1.6.2.6 NAME 'pcimRuleEnabled'
      DESC 'An integer indicating whether a policy rule is
            administratively enabled (value=1), disabled
```

                    (value=2), or enabled for debug (value=3).'
            EQUALITY integerMatch
            ORDERING integerOrderingMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
            SINGLE-VALUE
)

   Note: All other values for the pcimRuleEnabled attribute are
   considered errors, and the administrator SHOULD treat this rule as
   being disabled if an invalid value is found.

   The pcimRuleConditionListType attribute is used to indicate whether
   the list of policy conditions associated with this policy rule is in
   disjunctive normal form (DNF, value=1) or conjunctive normal form
   (CNF, value=2). It is defined as follows:

      (1.3.6.1.1.6.2.7 NAME 'pcimRuleConditionListType'
            DESC 'A value of 1 means that this policy rule is in
                   disjunctive normal form; a value of 2 means that this
                   policy rule is in conjunctive normal form.'
            EQUALITY integerMatch
            ORDERING integerOrderingMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
            SINGLE-VALUE
)

   Note: any value other than 1 or 2 for the pcimRuleConditionListType
   attribute is considered an error. Administrators SHOULD treat this
   rule as being disabled if an invalid value is found, since it is
   unclear how to structure the condition list.

   The pcimRuleConditionList attribute is a multi-valued attribute that
   is used to realize the policyRuleInPolicyCondition association
   defined in [1]. It contains a set of DNs of
   pcimRuleConditionAssociation entries representing associations
   between this policy rule and its conditions. No order is implied.
   It is defined as follows:

      (1.3.6.1.1.6.2.8 NAME 'pcimRuleConditionList'
            DESC 'Unordered set of DNs of pcimRuleConditionAssociation
                   entries representing associations between this policy
                   rule and its conditions.'
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)

The pcimRuleActionList attribute is a multi-valued attribute that is
used to realize the policyRuleInPolicyAction association defined in
[1]. It contains a set of DNs of pcimRuleActionAssociation entries
representing associations between this policy rule and its actions.
No order is implied. It is defined as follows:

```
( 1.3.6.1.1.6.2.9 NAME 'pcimRuleActionList'
      DESC 'Unordered set of DNs of pcimRuleActionAssociation
            entries representing associations between this policy
            rule and its actions.'
      EQUALITY distinguishedNameMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)
```

The pcimRuleValidityPeriodList attribute is a multi-valued attribute
that is used to realize the pcimRuleValidityPeriod association that
is defined in [1]. It contains a set of DNs of
pcimRuleValidityAssociation entries that determine when the pcimRule
is scheduled to be active or inactive. No order is implied. It is
defined as follows:

```
( 1.3.6.1.1.6.2.10 NAME 'pcimRuleValidityPeriodList'
      DESC 'Unordered set of DNs of pcimRuleValidityAssociation
            entries that determine when the pcimRule is scheduled
            to be active or inactive.'
      EQUALITY distinguishedNameMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)
```

The pcimRuleUsage attribute is a free-form string providing
guidelines on how this policy should be used. It is defined as
follows:

```
( 1.3.6.1.1.6.2.11 NAME 'pcimRuleUsage'
      DESC 'This attribute is a free-form sting providing
            guidelines on how this policy should be used.'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
      SINGLE-VALUE
)
```

The pcimRulePriority attribute is a non-negative integer that is used
to prioritize this pcimRule relative to other pcimRules. A larger
value indicates a higher priority. It is defined as follows:

```
( 1.3.6.1.1.6.2.12 NAME 'pcimRulePriority'
      DESC 'A non-negative integer for prioritizing this
            pcimRule relative to other pcimRules.  A larger
            value indicates a higher priority.'
      EQUALITY integerMatch
      ORDERING integerOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
      SINGLE-VALUE
)
```

Note: if the value of the pcimRulePriority field is 0, then it SHOULD
be treated as "don't care". On the other hand, if the value is
negative, then it SHOULD be treated as an error and Administrators
SHOULD treat this rule as being disabled.

The pcimRuleMandatory attribute is a Boolean attribute that, if TRUE,
indicates that for this policy rule, the evaluation of its conditions
and execution of its actions (if the condition is satisfied) is
required. If it is FALSE, then the evaluation of its conditions and
execution of its actions (if the condition is satisfied) is not
required. This attribute is defined as follows:

```
( 1.3.6.1.1.6.2.13 NAME 'pcimRuleMandatory'
      DESC 'If TRUE, indicates that for this policy rule, the
            evaluation of its conditions and execution of its
            actions (if the condition is satisfied) is required.'
      EQUALITY booleanMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
      SINGLE-VALUE
)
```

The pcimRuleSequencedActions attribute is an integer enumeration that
is used to indicate that the ordering of actions defined by the
pcimActionOrder attribute is either mandatory(value=1),
recommended(value=2), or dontCare(value=3). It is defined as
follows:

```
( 1.3.6.1.1.6.2.14 NAME 'pcimRuleSequencedActions'
      DESC 'An integer enumeration indicating that the ordering of
            actions defined by the pcimActionOrder attribute is
            mandatory(1), recommended(2), or dontCare(3).'
      EQUALITY integerMatch
      ORDERING integerOrderingMatch
```

```
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
            SINGLE-VALUE
   )
```

Note: if the value of pcimRulesSequencedActions field is not one of
these three values, then Administrators SHOULD treat this rule as
being disabled.

The pcimRoles attribute represents the policyRoles property of [1].
Each value of this attribute represents a role-combination, which is
a string of the form:
    <RoleName>[&&<RoleName>]* where the individual role names appear
in alphabetical order according to the collating sequence for UCS-2.
This attribute is defined as follows:

```
   ( 1.3.6.1.1.6.2.15 NAME 'pcimRoles'
         DESC 'Each value of this attribute represents a role-
               combination.'
         EQUALITY caseIgnoreMatch
         ORDERING caseIgnoreOrderingMatch
         SUBSTR caseIgnoreSubstringsMatch
         SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   )
```

Note: if the value of the pcimRoles attribute does not conform to the
format "<RoleName>[&&<RoleName>]*" (see Section 6.3.7 of [1]), then
this attribute is malformed and its policy rule SHOULD be treated as
being disabled.

The two subclasses of the pcimRule class are defined as follows.
First, the pcimRuleAuxClass is an auxiliary class for representing
the "If Condition then Action" semantics associated with a policy
rule. Its class definition is as follows:

```
   ( 1.3.6.1.1.6.1.6 NAME 'pcimRuleAuxClass'
         DESC 'An auxiliary class for representing the "If Condition
               then Action" semantics associated with a policy rule.'
         SUP pcimRule
         AUXILIARY
   )
```

The pcimRuleInstance is a structural class for representing the "If
Condition then Action" semantics associated with a policy rule. Its
class definition is as follows:

```
   ( 1.3.6.1.1.6.1.7 NAME 'pcimRuleInstance'
         DESC 'A structural class for representing the "If Condition
               then Action" semantics associated with a policy rule.'
```

```
                SUP pcimRule
                STRUCTURAL
        )
```

A DIT content rule could be written to enable an instance of
pcimRuleInstance to have attached to it either references to one or
more policy conditions (using pcimConditionAuxClass) or references to
one or more policy actions (using pcimActionAuxClass). This would be
used to formalize the semantics of the PolicyRule class [1]. Since
these semantics do not include specifying any properties of the
PolicyRule class, the content rule would not need to specify any
attributes.

Similarly, three separate DIT structure rules could be written, each
of which would refer to a specific name form that identified one of
its three possible naming attributes (i.e., pcimRuleName, cn, and
orderedCIMKeys). This structure rule SHOULD include a
superiorStructureRule (see Note 2 at the beginning of section 5).
The three name forms referenced by the three structure rules would
each define one of the three naming attributes.

5.4. The Class pcimRuleConditionAssociation

This class contains attributes to represent the properties of the
PCIM's PolicyConditionInPolicyRule association. Instances of this
class are related to an instance of pcimRule via DIT containment.
The policy conditions themselves are represented by auxiliary
subclasses of the auxiliary class pcimConditionAuxClass. These
auxiliary classes are attached directly to instances of
pcimRuleConditionAssociation for rule-specific policy conditions.
For a reusable policy condition, the policyCondition auxiliary
subclass is attached to an instance of the class pcimPolicyInstance
(which is presumably associated with a pcimRepository by DIT
containment), and the policyConditionDN attribute (of this class) is
used to reference the reusable policyCondition instance.

The class definition is as follows:

```
   ( 1.3.6.1.1.6.1.8 NAME 'pcimRuleConditionAssociation'
        DESC 'This class contains attributes characterizing the
              relationship between a policy rule and one of its
              policy conditions.'
        SUP pcimPolicy
        MUST ( pcimConditionGroupNumber $ pcimConditionNegated )
        MAY ( pcimConditionName $ pcimConditionDN )
   )
```

The attributes of this class are defined as follows.

The pcimConditionGroupNumber attribute is a non-negative integer. It
is used to identify the group to which the condition referenced by
this association is assigned. This attribute is defined as follows:

```
( 1.3.6.1.1.6.2.16
        NAME 'pcimConditionGroupNumber'
        DESC 'The number of the group to which a policy condition
              belongs.  This is used to form the DNF or CNF
              expression associated with a policy rule.'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE
)
```

Note that this number is non-negative. A negative value for this
attribute is invalid, and any policy rule that refers to an invalid
entry SHOULD be treated as being disabled.

The pcimConditionNegated attribute is a Boolean attribute that
indicates whether this policy condition is to be negated or not. If
it is TRUE (FALSE), it indicates that a policy condition IS (IS NOT)
negated in the DNF or CNF expression associated with a policy rule.
This attribute is defined as follows:

```
( 1.3.6.1.1.6.2.17
        NAME 'pcimConditionNegated'
        DESC 'If TRUE (FALSE), it indicates that a policy condition
              IS (IS NOT) negated in the DNF or CNF expression
              associated with a policy rule.'
        EQUALITY booleanMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        SINGLE-VALUE
)
```

The pcimConditionName is a user-friendly name for identifying this
policy condition, and may be used as a naming attribute if desired.
This attribute is defined as follows:

```
( 1.3.6.1.1.6.2.18
        NAME 'pcimConditionName'
        DESC 'A user-friendly name for a policy condition.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
```

```
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE
    )
```

The pcimConditionDN attribute is a DN that references an instance of
a reusable policy condition. This attribute is defined as follows:

```
   ( 1.3.6.1.1.6.2.19
          NAME 'pcimConditionDN'
          DESC 'A DN that references an instance of a reusable policy
               condition.'
          EQUALITY distinguishedNameMatch
          SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
          SINGLE-VALUE
   )
```

A DIT content rule could be written to enable an instance of
pcimRuleConditionAssociation to have attached to it an instance of
the auxiliary class pcimConditionAuxClass, or one of its subclasses.
This would be used to formalize the semantics of the
PolicyConditionInPolicyRule association. Specifically, this would be
used to represent a rule-specific policy condition [1].
Similarly, three separate DIT structure rules could be written. Each
of these DIT structure rules would refer to a specific name form that
defined two important semantics. First, each name form would
identify one of the three possible naming attributes (i.e.,
pcimConditionName, cn, and orderedCIMKeys) for the
pcimRuleConditionAssociation object class. Second, each name form
would require that an instance of the pcimRuleConditionAssociation
class have as its superior an instance of the pcimRule class. This
structure rule SHOULD also include a superiorStructureRule (see Note
2 at the beginning of section 5).

5.5. The Class pcimRuleValidityAssociation

The policyRuleValidityPeriod aggregation is mapped to the PCLS
pcimRuleValidityAssociation class. This class represents the
scheduled activation and deactivation of a policy rule by binding the
definition of times that the policy is active to the policy rule
itself. The "scheduled" times are either identified through an
attached auxiliary class pcimTPCAuxClass, or are referenced through
its pcimTimePeriodConditionDN attribute.

This class is defined as follows:

```
   ( 1.3.6.1.1.6.1.9 NAME 'pcimRuleValidityAssociation'
         DESC 'This defines the scheduled activation or deactivation
              of a policy rule.'
```

```
        SUP pcimPolicy
        STRUCTURAL
        MAY ( pcimValidityConditionName $ pcimTimePeriodConditionDN )
    )
```

The attributes of this class are defined as follows:

The pcimValidityConditionName attribute is used to define a
user-friendly name of this condition, and may be used as a naming
attribute if desired. This attribute is defined as follows:

```
    ( 1.3.6.1.1.6.2.20
            NAME 'pcimValidityConditionName'
            DESC 'A user-friendly name for identifying an instance of
                  a pcimRuleValidityAssociation entry.'
            EQUALITY caseIgnoreMatch
            ORDERING caseIgnoreOrderingMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE
    )
```

The pcimTimePeriodConditionDN attribute is a DN that references a
reusable time period condition. It is defined as follows:

```
    ( 1.3.6.1.1.6.2.21
            NAME 'pcimTimePeriodConditionDN'
             DESC 'A reference to a reusable policy time period
                   condition.'
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
            SINGLE-VALUE
    )
```

A DIT content rule could be written to enable an instance of
pcimRuleValidityAssociation to have attached to it an instance of the
auxiliary class pcimTPCAuxClass, or one of its subclasses. This
would be used to formalize the semantics of the
PolicyRuleValidityPeriod aggregation [1].

Similarly, three separate DIT structure rules could be written. Each
of these DIT structure rules would refer to a specific name form that
defined two important semantics. First, each name form would
identify one of the three possible naming attributes (i.e.,
pcimValidityConditionName, cn, and orderedCIMKeys) for the
pcimRuleValidityAssociation object class. Second, each name form
would require that an instance of the pcimRuleValidityAssociation
class have as its superior an instance of the pcimRule class. This

   structure rule SHOULD also include a superiorStructureRule (see Note
   2 at the beginning of section 5).

5.6. The Class pcimRuleActionAssociation

   This class contains an attribute to represent the one property of the
   PCIM PolicyActionInPolicyRule association, ActionOrder. This
   property is used to specify an order for executing the actions
   associated with a policy rule. Instances of this class are related
   to an instance of pcimRule via DIT containment. The actions
   themselves are represented by auxiliary subclasses of the auxiliary
   class pcimActionAuxClass.

   These auxiliary classes are attached directly to instances of
   pcimRuleActionAssociation for rule-specific policy actions. For a
   reusable policy action, the pcimAction auxiliary subclass is attached
   to an instance of the class pcimPolicyInstance (which is presumably
   associated with a pcimRepository by DIT containment), and the
   pcimActionDN attribute (of this class) is used to reference the
   reusable pcimCondition instance.

   The class definition is as follows:

      (1.3.6.1.1.6.1.10 NAME 'pcimRuleActionAssociation'
            DESC 'This class contains attributes characterizing the
                  relationship between a policy rule and one of its
                  policy actions.'
            SUP pcimPolicy
            MUST (pcimActionOrder)
            MAY (pcimActionName $ pcimActionDN)
)

   The pcimActionName attribute is used to define a user-friendly name
   of this action, and may be used as a naming attribute if desired.
   This attribute is defined as follows:

      (1.3.6.1.1.6.2.22
            NAME 'pcimActionName'
            DESC 'A user-friendly name for a policy action.'
            EQUALITY caseIgnoreMatch
            ORDERING caseIgnoreOrderingMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE
)

The pcimActionOrder attribute is an unsigned integer that is used to
indicate the relative position of an action in a sequence of actions
that are associated with a given policy rule. When this number is
positive, it indicates a place in the sequence of actions to be
performed, with smaller values indicating earlier positions in the
sequence. If the value is zero, then this indicates that the order
is irrelevant. Note that if two or more actions have the same
non-zero value, they may be performed in any order as long as they
are each performed in the correct place in the overall sequence of
actions. This attribute is defined as follows:

```
( 1.3.6.1.1.6.2.23
      NAME 'pcimActionOrder'
      DESC 'An integer indicating the relative order of an action
            in the context of a policy rule.'
      EQUALITY integerMatch
      ORDERING integerOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
      SINGLE-VALUE
)
```

Note: if the value of the pcimActionOrder field is negative, then it
SHOULD be treated as an error and any policy rule that refers to such
an entry SHOULD be treated as being disabled.

The pcimActionDN attribute is a DN that references a reusable policy
action. It is defined as follows:

```
( 1.3.6.1.1.6.2.24
      NAME 'pcimActionDN'
      DESC 'A DN that references a reusable policy action.'
      EQUALITY distinguishedNameMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
      SINGLE-VALUE
)
```

A DIT content rule could be written to enable an instance of
pcimRuleActionAssociation to have attached to it an instance of the
auxiliary class pcimActionAuxClass, or one of its subclasses. This
would be used to formalize the semantics of the
PolicyActionInPolicyRule association. Specifically, this would be
used to represent a rule-specific policy action [1].

Similarly, three separate DIT structure rules could be written. Each
of these DIT structure rules would refer to a specific name form that
defined two important semantics. First, each name form would
identify one of the three possible naming attributes (i.e.,
pcimActionName, cn, and orderedCIMKeys) for the

pcimRuleActionAssociation object class. Second, each name form would
require that an instance of the pcimRuleActionAssociation class have
as its superior an instance of the pcimRule class. This structure
rule should also include a superiorStructureRule (see Note 2 at the
beginning of section 5).

5.7. The Auxiliary Class pcimConditionAuxClass

   The purpose of a policy condition is to determine whether or not the
   set of actions (contained in the pcimRule that the condition applies
   to) should be executed or not. This class defines the basic
   organizational semantics of a policy condition, as specified in [1].
   Subclasses of this auxiliary class can be attached to instances of
   three other classes in the PCLS. When a subclass of this class is
   attached to an instance of pcimRuleConditionAssociation, or to an
   instance of pcimRule, it represents a rule-specific policy condition.
   When a subclass of this class is attached to an instance of
   pcimPolicyInstance, it represents a reusable policy condition.

   Since all of the classes to which subclasses of this auxiliary class
   may be attached are derived from the pcimPolicy class, the attributes
   of pcimPolicy will already be defined for the entries to which these
   subclasses attach. Thus, this class is derived directly from "top".

   The class definition is as follows:

      (1.3.6.1.1.6.1.11 NAME 'pcimConditionAuxClass'
           DESC 'A class representing a condition to be evaluated in
                conjunction with a policy rule.'
           SUP top
           AUXILIARY
)

5.8. The Auxiliary Class pcimTPCAuxClass

   The PCIM defines a time period class, PolicyTimePeriodCondition, to
   provide a means of representing the time periods during which a
   policy rule is valid, i.e., active. It also defines an aggregation,
   PolicyRuleValidityPeriod, so that time periods can be associated with
   a PolicyRule. The LDAP mapping also provides two classes, one for
   the time condition itself, and one for the aggregation.

   In the PCIM, the time period class is named
   PolicyTimePeriodCondition. However, the resulting name of the
   auxiliary class in this mapping (pcimTimePeriodConditionAuxClass)
   exceeds the length of a name that some directories can store.
   Therefore, the name has been shortened to pcimTPCAuxClass.

The class definition is as follows:

```
( 1.3.6.1.1.6.1.12 NAME 'pcimTPCAuxClass'
        DESC 'This provides the capability of enabling or disabling
              a policy rule according to a predetermined schedule.'
        SUP pcimConditionAuxClass
        AUXILIARY
        MAY ( pcimTPCTime $ pcimTPCMonthOfYearMask $
              pcimTPCDayOfMonthMask $ pcimTPCDayOfWeekMask $
              pcimTPCTimeOfDayMask $ pcimTPCLocalOrUtcTime )
)
```

The attributes of the pcimTPCAuxClass are defined as follows.

The pcimTPCTime attribute represents the time period that a policy
rule is enabled for. This attribute is defined as a string in [1]
with a special format which defines a time period with a starting
date and an ending date separated by a forward slash ("/"), as
follows:

```
yyyymmddThhmmss/yyyymmddThhmmss
```

where the first date and time may be replaced with the string
"THISANDPRIOR" or the second date and time may be replaced with the
string "THISANDFUTURE". This attribute is defined as follows:

```
( 1.3.6.1.1.6.2.25
        NAME 'pcimTPCTime'
        DESC 'The start and end times on which a policy rule is
              valid.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.44
        SINGLE-VALUE
)
```

The value of this attribute SHOULD be checked against its defined
format ("yyyymmddThhmmss/yyyymmddThhmmss", where the first and second
date strings may be replaced with the strings "THISANDPRIOR" and
"THISANDFUTURE"). If the value of this attribute does not conform to
this syntax, then this SHOULD be considered an error and the policy
rule SHOULD be treated as being disabled.

The next four attributes (pcimTPCMonthOfYearMask,
pcimTPCDayOfMonthMask, pcimTPCDayOfWeekMask, and
pcimTPCTimeOfDayMask) are all defined as octet strings in [1].
However, the semantics of each of these attributes are contained in

bit strings of various fixed lengths. Therefore, the PCLS uses a
syntax of Bit String to represent each of them. The definition of
these four attributes are as follows.

The pcimTPCMonthOfYearMask attribute defines a 12-bit mask
identifying the months of the year in which a policy rule is valid.
The format is a bit string of length 12, representing the months of
the year from January through December. The definition of this
attribute is as follows:

```
( 1.3.6.1.1.6.2.26
        NAME 'pcimTPCMonthOfYearMask'
        DESC 'This identifies the valid months of the year for a
              policy rule using a 12-bit string that represents the
              months of the year from January through December.'
        EQUALITY bitStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.6
        SINGLE-VALUE
)
```

The value of this attribute SHOULD be checked against its defined
format. If the value of this attribute does not conform to this
syntax, then this SHOULD be considered an error and the policy rule
SHOULD be treated as being disabled.

The pcimTPCMonthOfDayMask attribute defines a mask identifying the
days of the month on which a policy rule is valid. The format is a
bit string of length 62. The first 31 positions represent the days
of the month in ascending order, from day 1 to day 31. The next 31
positions represent the days of the month in descending order, from
the last day to the day 31 days from the end. The definition of this
attribute is as follows:

```
( 1.3.6.1.1.6.2.27
        NAME 'pcimTPCDayOfMonthMask'
        DESC 'This identifies the valid days of the month for a
              policy rule using a 62-bit string. The first 31
              positions represent the days of the month in ascending
              order, and the next 31 positions represent the days of
              the month in descending order.'
        EQUALITY bitStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.6
        SINGLE-VALUE
)
```

The value of this attribute SHOULD be checked against its defined
format. If the value of this attribute does not conform to this
syntax, then this SHOULD be considered an error and the policy rule
SHOULD be treated as being disabled.

The pcimTPCDayOfWeekMask attribute defines a mask identifying the
days of the week on which a policy rule is valid. The format is a
bit string of length 7, representing the days of the week from Sunday
through Saturday. The definition of this attribute is as follows:

```
   ( 1.3.6.1.1.6.2.28
           NAME 'pcimTPCDayOfWeekMask'
           DESC 'This identifies the valid days of the week for a
                   policy rule using a 7-bit string. This represents
                   the days of the week from Sunday through Saturday.'
           EQUALITY bitStringMatch
           SYNTAX 1.3.6.1.4.1.1466.115.121.1.6
           SINGLE-VALUE
   )
```

The value of this attribute SHOULD be checked against its defined
format. If the value of this attribute does not conform to this
syntax, then this SHOULD be considered an error and the policy rule
SHOULD be treated as being disabled.

The pcimTPCTimeOfDayMask attribute defines the range of times at
which a policy rule is valid. If the second time is earlier than the
first, then the interval spans midnight. The format of the string is
Thhmmss/Thhmmss. The definition of this attribute is as follows:

```
   ( 1.3.6.1.1.6.2.29
           NAME 'pcimTPCTimeOfDayMask'
           DESC 'This identifies the valid range of times for a policy
                   using the format Thhmmss/Thhmmss.'
           EQUALITY caseIgnoreMatch
           ORDERING caseIgnoreOrderingMatch
           SUBSTR caseIgnoreSubstringsMatch
           SYNTAX 1.3.6.1.4.1.1466.115.121.1.44
           SINGLE-VALUE
   )
```

The value of this attribute SHOULD be checked against its defined
format. If the value of this attribute does not conform to this
syntax, then this SHOULD be considered an error and the policy rule
SHOULD be treated as being disabled.

Finally, the pcimTPCLocalOrUtcTime attribute is used to choose
between local or UTC time representation. This is mapped as a simple
integer syntax, with the value of 1 representing local time and the
value of 2 representing UTC time. The definition of this attribute
is as follows:

```
( 1.3.6.1.1.6.2.30
        NAME 'pcimTPCLocalOrUtcTime'
        DESC 'This defines whether the times in this instance
               represent local (value=1) times or UTC (value=2)
               times.'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE
)
```

Note: if the value of the pcimTPCLocalOrUtcTime is not 1 or 2, then
this SHOULD be considered an error and the policy rule SHOULD be
disabled. If the attribute is not present at all, then all times are
interpreted as if it were present with the value 2, that is, UTC
time.

5.9. The Auxiliary Class pcimConditionVendorAuxClass

This class provides a general extension mechanism for representing
policy conditions that have not been modeled with specific
properties. Instead, its two properties are used to define the
content and format of the condition, as explained below. This class
is intended for vendor-specific extensions that are not amenable to
using pcimCondition; standardized extensions SHOULD NOT use this
class.

The class definition is as follows:

```
( 1.3.6.1.1.6.1.13 NAME 'pcimConditionVendorAuxClass'
        DESC 'A class that defines a registered means to describe a
               policy condition.'
        SUP pcimConditionAuxClass
        AUXILIARY
        MAY ( pcimVendorConstraintData $
            pcimVendorConstraintEncoding )
)
```

The pcimVendorConstraintData attribute is a multi-valued attribute.
It provides a general mechanism for representing policy conditions
that have not been modeled as specific attributes. This information
is encoded in a set of octet strings. The format of the octet

strings is identified by the OID stored in the
pcimVendorConstraintEncoding attribute. This attribute is defined as
follows:

```
( 1.3.6.1.1.6.2.31
        NAME 'pcimVendorConstraintData'
        DESC 'Mechanism for representing constraints that have not
              been modeled as specific attributes.  Their format is
              identified by the OID stored in the attribute
              pcimVendorConstraintEncoding.'
        EQUALITY octetStringMatch
        ORDERING octetStringOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
)
```

The pcimVendorConstraintEncoding attribute is used to identify the
format and semantics for the pcimVendorConstraintData attribute.
This attribute is defined as follows:

```
( 1.3.6.1.1.6.2.32
        NAME 'pcimVendorConstraintEncoding'
        DESC 'An OID identifying the format and semantics for the
              pcimVendorConstraintData for this instance.'
        EQUALITY objectIdentifierMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
        SINGLE-VALUE
)
```

5.10. The Auxiliary Class pcimActionAuxClass

The purpose of a policy action is to execute one or more operations
that will affect network traffic and/or systems, devices, etc. in
order to achieve a desired policy state. This class is used to
represent an action to be performed as a result of a policy rule
whose condition clause was satisfied.

Subclasses of this auxiliary class can be attached to instances of
three other classes in the PCLS. When a subclass of this class is
attached to an instance of pcimRuleActionAssociation, or to an
instance of pcimRule, it represents a rule-specific policy action.
When a subclass of this class is attached to an instance of
pcimPolicyInstance, it represents a reusable policy action.

Since all of the classes to which subclasses of this auxiliary class
may be attached are derived from the pcimPolicy class, the attributes
of the pcimPolicy class will already be defined for the entries to
which these subclasses attach. Thus, this class is derived directly
from "top".

The class definition is as follows:

```
( 1.3.6.1.1.6.1.14 NAME 'pcimActionAuxClass'
      DESC 'A class representing an action to be performed as a
            result of a policy rule.'
      SUP top
      AUXILIARY
)
```

5.11. The Auxiliary Class pcimActionVendorAuxClass

   The purpose of this class is to provide a general extension mechanism
   for representing policy actions that have not been modeled with
   specific properties. Instead, its two properties are used to define
   the content and format of the action, as explained below.

   As its name suggests, this class is intended for vendor-specific
   extensions that are not amenable to using the standard pcimAction
   class. Standardized extensions SHOULD NOT use this class.

   The class definition is as follows:

```
( 1.3.6.1.1.6.1.15 NAME 'pcimActionVendorAuxClass'
      DESC 'A class that defines a registered means to describe a
            policy action.'
      SUP pcimActionAuxClass
      AUXILIARY
      MAY ( pcimVendorActionData $ pcimVendorActionEncoding )
)
```

   The pcimVendorActionData attribute is a multi-valued attribute. It
   provides a general mechanism for representing policy actions that
   have not been modeled as specific attributes. This information is
   encoded in a set of octet strings. The format of the octet strings
   is identified by the OID stored in the pcimVendorActionEncoding
   attribute. This attribute is defined as follows:

```
( 1.3.6.1.1.6.2.33
      NAME 'pcimVendorActionData'
      DESC ' Mechanism for representing policy actions that have
            not been modeled as specific attributes.  Their
            format is identified by the OID stored in the
            attribute pcimVendorActionEncoding.'
      EQUALITY octetStringMatch
      ORDERING octetStringOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
)
```

The pcimVendorActionEncoding attribute is used to identify the format
and semantics for the pcimVendorActionData attribute. This attribute
is defined as follows:

```
( 1.3.6.1.1.6.2.34
        NAME 'pcimVendorActionEncoding'
        DESC 'An OID identifying the format and semantics for the
              pcimVendorActionData attribute of this instance.'
        EQUALITY objectIdentifierMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
        SINGLE-VALUE
)
```

5.12. The Class pcimPolicyInstance

   This class is not defined in the PCIM. Its role is to serve as a
   structural class to which auxiliary classes representing policy
   information are attached when the information is reusable. For
   auxiliary classes representing policy conditions and policy actions,
   there are alternative structural classes that may be used. See
   Section 4.4 for a complete discussion of reusable policy conditions
   and actions, and of the role that this class plays in how they are
   represented.

   The class definition is as follows:

```
( 1.3.6.1.1.6.1.16 NAME 'pcimPolicyInstance'
        DESC 'A structural class to which aux classes containing
              reusable policy information can be attached.'
        SUP pcimPolicy
        MAY ( pcimPolicyInstanceName )
)
```

   The pcimPolicyInstanceName attribute is used to define a
   user-friendly name of this class, and may be used as a naming
   attribute if desired. It is defined as follows:

```
( 1.3.6.1.1.6.2.35 NAME 'pcimPolicyInstanceName'
        DESC 'The user-friendly name of this policy instance.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE
)
```

A DIT content rule could be written to enable an instance of
pcimPolicyInstance to have attached to it either instances of one or
more of the auxiliary object classes pcimConditionAuxClass and
pcimActionAuxClass. Since these semantics do not include specifying
any properties, the content rule would not need to specify any
attributes. Note that other content rules could be defined to enable
other policy-related auxiliary classes to be attached to
pcimPolicyInstance.

Similarly, three separate DIT structure rules could be written. Each
of these DIT structure rules would refer to a specific name form that
defined two important semantics. First, each name form would
identify one of the three possible naming attributes (i.e.,
pcimPolicyInstanceName, cn, and orderedCIMKeys) for this object
class. Second, each name form would require that an instance of the
pcimPolicyInstance class have as its superior an instance of the
pcimRepository class. This structure rule SHOULD also include a
superiorStructureRule (see Note 2 at the beginning of section 5).

5.13. The Auxiliary Class pcimElementAuxClass

   This class introduces no additional attributes, beyond those defined
   in the class pcimPolicy from which it is derived. Its role is to
   "tag" an instance of a class defined outside the realm of policy
   information as represented by PCIM as being nevertheless relevant to
   a policy specification. This tagging can potentially take place at
   two levels:

   -   Every instance to which pcimElementAuxClass is attached becomes
       an instance of the class pcimPolicy, since pcimElementAuxClass is
       a subclass of pcimPolicy. Searching for object
       class="pcimPolicy" will return the instance. (As noted earlier,
       this approach does NOT work for some directory implementations.
       To accommodate these implementations, policy-related entries
       SHOULD be tagged with the pcimKeyword "POLICY".)

   -   With the pcimKeywords attribute that it inherits from pcimPolicy,
       an instance to which pcimElementAuxClass is attached can be
       tagged as being relevant to a particular type or category of
       policy information, using standard keywords,
       administrator-defined keywords, or both.

   The class definition is as follows:

     (1.3.6.1.1.6.1.17 NAME 'pcimElementAuxClass'
           DESC 'An auxiliary class used to tag instances of classes
                 defined outside the realm of policy as relevant to a
                 particular policy specification.'

```
                SUP pcimPolicy
                AUXILIARY
        )
```

5.14. The Three Policy Repository Classes

   These classes provide a container for reusable policy information,
   such as reusable policy conditions and/or reusable policy actions.
   This document is concerned with mapping just the properties that
   appear in these classes. Conceptually, this may be thought of as a
   special location in the DIT where policy information may reside.
   Since pcimRepository is derived from the class dlm1AdminDomain
   defined in reference [6], this specification has a normative
   dependency on that element of reference [6] (as well as on its entire
   derivation hierarchy, which also appears in reference [6]). To
   maximize flexibility, the pcimRepository class is defined as
   abstract. A subclass pcimRepositoryAuxClass provides for auxiliary
   attachment to another entry, while a structural subclass
   pcimRepositoryInstance is available to represent a policy repository
   as a standalone entry.

   The definition for the pcimRepository class is as follows:

```
        ( 1.3.6.1.1.6.1.18 NAME 'pcimRepository'
                DESC 'A container for reusable policy information.'
                SUP dlm1AdminDomain
                ABSTRACT
                MAY ( pcimRepositoryName )
        )
```

   The pcimRepositoryName attribute is used to define a user-friendly
   name of this class, and may be used as a naming attribute if desired.
   It is defined as follows:

```
        ( 1.3.6.1.1.6.2.36 NAME 'pcimRepositoryName'
                DESC 'The user-friendly name of this policy repository.'
                EQUALITY caseIgnoreMatch
                ORDERING caseIgnoreOrderingMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
                SINGLE-VALUE
        )
```

   The two subclasses of pcimRepository are defined as follows. First,
   the pcimRepositoryAuxClass is an auxiliary class that can be used to
   aggregate reusable policy information. It is defined as follows:

      (1.3.6.1.1.6.1.19 NAME 'pcimRepositoryAuxClass'
            DESC 'An auxiliary class that can be used to aggregate
                  reusable policy information.'
            SUP pcimRepository
            AUXILIARY
)

   In cases where structural classes are needed instead of an auxiliary
   class, the pcimRepositoryInstance class is a structural class that
   can be used to aggregate reusable policy information. It is defined
   as follows:

      (1.3.6.1.1.6.1.20 NAME 'pcimRepositoryInstance'
            DESC 'A structural class that can be used to aggregate
                  reusable policy information.'
            SUP pcimRepository
            STRUCTURAL
)

   Three separate DIT structure rules could be written for this class.
   Each of these DIT structure rules would refer to a specific name form
   that enabled an instance of the pcimRepository class to be named
   under any superior using one of the three possible naming attributes
   (i.e., pcimRepositoryName, cn, and orderedCIMKeys). This structure
   rule SHOULD also include a superiorStructureRule (see Note 2 at the
   beginning of section 5).

5.15. The Auxiliary Class pcimSubtreesPtrAuxClass

   This auxiliary class provides a single, multi-valued attribute that
   references a set of objects that are at the root of DIT subtrees
   containing policy-related information. By attaching this attribute
   to instances of various other classes, a policy administrator has a
   flexible way of providing an entry point into the directory that
   allows a client to locate and retrieve the policy information
   relevant to it.

   It is intended that these entries are placed in the DIT such that
   well-known DNs can be used to reference a well-known structural entry
   that has the pcimSubtreesPtrAuxClass attached to it. In effect, this
   defines a set of entry points. Each of these entry points can
   contain and/or reference all related policy entries for

any well-known policy domains. The pcimSubtreesPtrAuxClass functions
as a tag to identify portions of the DIT that contain policy
information.

This object does not provide the semantic linkages between individual
policy objects, such as those between a policy group and the policy
rules that belong to it. Its only role is to enable efficient bulk
retrieval of policy-related objects, as described in Section 4.5.

Once the objects have been retrieved, a directory client can
determine the semantic linkages by following references contained in
multi-valued attributes, such as pcimRulesAuxContainedSet.

Since policy-related objects will often be included in the DIT
subtree beneath an object to which this auxiliary class is attached,
a client SHOULD request the policy-related objects from the subtree
under the object with these references at the same time that it
requests the references themselves.

Since clients are expected to behave in this way, the policy
administrator SHOULD make sure that this subtree does not contain so
many objects unrelated to policy that an initial search done in this
way results in a performance problem. The pcimSubtreesPtrAuxClass
SHOULD NOT be attached to the partition root for a large directory
partition containing a relatively few number of policy-related
objects along with a large number of objects unrelated to policy
(again, "policy" here refers to the PCIM, not the X.501, definition
and use of "policy"). A better approach would be to introduce a
container object immediately below the partition root, attach
pcimSubtreesPtrAuxClass to this container object, and then place all
of the policy-related objects in that subtree.

The class definition is as follows:

    (1.3.6.1.1.6.1.21 NAME 'pcimSubtreesPtrAuxClass'
          DESC 'An auxiliary class providing DN references to roots of
               DIT subtrees containing policy-related objects.'
          SUP top
          AUXILIARY
          MAY (pcimSubtreesAuxContainedSet)
)

The attribute pcimSubtreesAuxContainedSet provides an unordered set
of DN references to instances of one or more objects under which
policy-related information is present. The objects referenced may or
may not themselves contain policy-related information. The attribute
definition is as follows:

```
( 1.3.6.1.1.6.2.37
        NAME 'pcimSubtreesAuxContainedSet'
        DESC 'DNs of objects that serve as roots for DIT subtrees
              containing policy-related objects.'
        EQUALITY distinguishedNameMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)
```

Note that the cn attribute does NOT need to be defined for this
class. This is because an auxiliary class is used as a means to
collect common attributes and treat them as properties of an object.
A good analogy is a #include file, except that since an auxiliary
class is a class, all the benefits of a class (e.g., inheritance) can
be applied to an auxiliary class.

5.16. The Auxiliary Class pcimGroupContainmentAuxClass

   This auxiliary class provides a single, multi-valued attribute that
   references a set of pcimGroups. By attaching this attribute to
   instances of various other classes, a policy administrator has a
   flexible way of providing an entry point into the directory that
   allows a client to locate and retrieve the pcimGroups relevant to it.

   As is the case with pcimRules, a policy administrator might have
   several different references to a pcimGroup in the overall directory
   structure. The pcimGroupContainmentAuxClass is the mechanism that
   makes it possible for the policy administrator to define all these
   different references.

   The class definition is as follows:

```
( 1.3.6.1.1.6.1.22 NAME 'pcimGroupContainmentAuxClass'
        DESC 'An auxiliary class used to bind pcimGroups to an
              appropriate container object.'
        SUP top
        AUXILIARY
        MAY ( pcimGroupsAuxContainedSet )
)
```

   The attribute pcimGroupsAuxContainedSet provides an unordered set of
   references to instances of one or more pcimGroups associated with the
   instance of a structural class to which this attribute has been
   appended.

   The attribute definition is as follows:

      (1.3.6.1.1.6.2.38
            NAME 'pcimGroupsAuxContainedSet'
            DESC 'DNs of pcimGroups associated in some way with the
                  instance to which this attribute has been appended.'
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)

   Note that the cn attribute does NOT have to be defined for this class
   for the same reasons as those given for the pcimSubtreesPtrAuxClass
   in section 5.15.

5.17. The Auxiliary Class pcimRuleContainmentAuxClass

   This auxiliary class provides a single, multi-valued attribute that
   references a set of pcimRules. By attaching this attribute to
   instances of various other classes, a policy administrator has a
   flexible way of providing an entry point into the directory that
   allows a client to locate and retrieve the pcimRules relevant to it.

   A policy administrator might have several different references to a
   pcimRule in the overall directory structure. For example, there
   might be references to all pcimRules for traffic originating in a
   particular subnet from a directory entry that represents that subnet.
   At the same time, there might be references to all pcimRules related
   to a particular DiffServ setting from an instance of a pcimGroup
   explicitly introduced as a container for DiffServ-related pcimRules.
   The pcimRuleContainmentAuxClass is the mechanism that makes it
   possible for the policy administrator to define all these separate
   references.

   The class definition is as follows:

      (1.3.6.1.1.6.1.23 NAME 'pcimRuleContainmentAuxClass'
            DESC 'An auxiliary class used to bind pcimRules to an
                  appropriate container object.'
            SUP top
            AUXILIARY
            MAY (pcimRulesAuxContainedSet)
)

The attribute pcimRulesAuxContainedSet provides an unordered set of
references to one or more instances of pcimRules associated with the
instance of a structural class to which this attribute has been
appended. The attribute definition is as follows:

```
( 1.3.6.1.1.6.2.39
        NAME 'pcimRulesAuxContainedSet'
        DESC 'DNs of pcimRules associated in some way with the
              instance to which this attribute has been appended.'
        EQUALITY distinguishedNameMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)
```

The cn attribute does NOT have to be defined for this class for the
same reasons as those given for the pcimSubtreesPtrAuxClass in
section 5.15.

6. Extending the Classes Defined in This Document

   The following subsections provide general guidance on how to create a
   domain-specific schema derived from this document, discuss how the
   vendor classes in the PCLS should be used, and explain how
   policyTimePeriodConditions are related to other policy conditions.

6.1. Subclassing pcimConditionAuxClass and pcimActionAuxClass

   In Section 4.4, there is a discussion of how, by representing policy
   conditions and policy actions as auxiliary classes in a schema, the
   flexibility is retained to instantiate a particular condition or
   action as either rule-specific or reusable. This flexibility is lost
   if a condition or action class is defined as structural rather than
   auxiliary. For standardized schemata, this document specifies that
   domain-specific information MUST be expressed in auxiliary subclasses
   of pcimConditionAuxClass and pcimActionAuxClass. It is RECOMMENDED
   that non-standardized schemata follow this practice as well.

6.2. Using the Vendor Policy Attributes

   As discussed Section 5.9, the attributes pcimVendorConstraintData and
   pcimVendorConstraintEncoding are included in the
   pcimConditionVendorAuxClass to provide a mechanism for representing
   vendor-specific policy conditions that are not amenable to being
   represented with the pcimCondition class (or its subclasses). The
   attributes pcimVendorActionData and pcimVendorActionEncoding in the
   pcimActionVendorAuxClass class play the same role with respect to
   actions. This enables interoperability between different vendors who
   could not otherwise interoperate.

For example, imagine a network composed of access devices from vendor
A, edge and core devices from vendor B, and a policy server from
vendor C. It is desirable for this policy server to be able to
configure and manage all of the devices from vendors A and B.
Unfortunately, these devices will in general have little in common
(e.g., different mechanisms, different ways for controlling those
mechanisms, different operating systems, different commands, and so
forth). The extension conditions provide a way for vendor-specific
commands to be encoded as octet strings, so that a single policy
server can commonly manage devices from different vendors.

6.3. Using Time Validity Periods

Time validity periods are defined as an auxiliary subclass of
pcimConditionAuxClass, called pcimTPCAuxClass. This is to allow
their inclusion in the AND/OR condition definitions for a pcimRule.
Care should be taken not to subclass pcimTPCAuxClass to add
domain-specific condition properties.

For example, it would be incorrect to add IPsec- or QoS-specific
condition properties to the pcimTPCAuxClass class, just because IPsec
or QoS includes time in its condition definition. The correct
subclassing would be to create IPsec or QoS-specific subclasses of
pcimConditionAuxClass and then combine instances of these
domain-specific condition classes with the appropriate validity
period criteria. This is accomplished using the AND/OR association
capabilities for policy conditions in pcimRules.

7. Security Considerations

The PCLS, presented in this document, provides a mapping of the
object-oriented model for describing policy information (PCIM) into a
data model that forms the basic framework for describing the
structure of policy data, in the case where the policy repository
takes the form of an LDAP-accessible directory.

PCLS is not intended to represent any particular system design or
implementation. PCLS is not directly useable in a real world system,
without the discipline-specific mappings that are works in progress
in the Policy Framework Working Group of the IETF.

These other derivative documents, which use PCIM and its
discipline-specific extensions as a base, will need to convey more
specific security considerations (refer to RFC 3060 for more
information.)

The reason that PCLS, as defined here, is not representative of any
real-world system, is that its object classes were designed to be
independent of any specific discipline, or policy domain. For
example, DiffServ and IPsec represent two different policy domains.
Each document that extends PCIM to one of these domains will derive
subclasses from the classes and relationships defined in PCIM, in
order to represent extensions of a generic model to cover specific
technical domains.

PCIM-derived documents will thus subclass the PCIM classes into
classes specific to each technical policy domain (QOS, IPsec, etc.),
which will, in turn, be mapped, to directory-specific schemata
consistent with the PCLS documented here.

Even though discipline-specific security requirements are not
appropriate for PCLS, specific security requirements MUST be defined
for each operational real-world application of PCIM. Just as there
will be a wide range of operational, real-world systems using PCIM,
there will also be a wide range of security requirements for these
systems. Some operational, real-world systems that are deployed
using PCLS may have extensive security requirements that impact
nearly all object classes utilized by such a system, while other
systems' security requirements might have very little impact.

The derivative documents, discussed above, will create the context
for applying operational, real-world, system-level security
requirements against the various models that derive from PCIM,
consistent with PCLS.

In some real-world scenarios, the values associated with certain
properties, within certain instantiated object classes, may represent
information associated with scarce, and/or costly (and therefore
valuable) resources. It may be the case that these values must not
be disclosed to, or manipulated by, unauthorized parties.

Since this document forms the basis for the representation of a
policy data model in a specific format (an LDAP-accessible
directory), it is herein appropriate to reference the data
model-specific tools and mechanisms that are available for achieving
the authentication and authorization implicit in a requirement that
restricts read and/or read- write access to these values stored in a
directory.

General LDAP security considerations apply, as documented in RFC 3377
[2]. LDAP-specific authentication and authorization tools and
mechanisms are found in the following standards track documents,
which are appropriate for application to the management of security
applied to policy data models stored in an LDAP-accessible directory:

   -   RFC 2829 (Authentication Methods for LDAP)
   -   RFC 2830 (Lightweight Directory Access Protocol (v3): Extension
       for Transport Layer Security)

Any identified security requirements that are not dealt with in the
appropriate discipline-specific information model documents, or in
this document, MUST be dealt with in the derivative data model
documents which are specific to each discipline.

8.   IANA Considerations

Refer to RFC 3383, "Internet Assigned Numbers Authority (IANA)
Considerations for the Lightweight Directory Access Protocol (LDAP)"
[16].

8.1. Object Identifiers

The IANA has registered an LDAP Object Identifier for use in this
technical specification according to the following template:

Subject: Request for LDAP OID Registration
Person & email address to contact for further information:
   Bob Moore (remoore@us.ibm.com)
Specification: RFC 3703
Author/Change Controller: IESG
Comments:
   The assigned OID will be used as a base for identifying
   a number of schema elements defined in this document.

IANA has assigned an OID of 1.3.6.1.1.6 with the name of pcimSchema
to this registration as recorded in the following registry:

   http://www.iana.org/assignments/smi-numbers

8.2. Object Identifier Descriptors

The IANA has registered the LDAP Descriptors used in this technical
specification as detailed in the following template:

Subject: Request for LDAP Descriptor Registration Update
Descriptor (short name): see comment
Object Identifier: see comment

      Person & email address to contact for further information:
         Bob Moore (remoore@us.ibm.com)
      Usage: see comment
      Specification: RFC 3703
      Author/Change Controller: IESG
      Comments:

      The following descriptors have been added:

      NAME   Type   OID
      --------------   ----   ------------
      pcimPolicy   O   1.3.6.1.1.6.1.1
      pcimGroup   O   1.3.6.1.1.6.1.2
      pcimGroupAuxClass   O   1.3.6.1.1.6.1.3
      pcimGroupInstance   O   1.3.6.1.1.6.1.4
      pcimRule   O   1.3.6.1.1.6.1.5
      pcimRuleAuxClass   O   1.3.6.1.1.6.1.6
      pcimRuleInstance   O   1.3.6.1.1.6.1.7
      pcimRuleConditionAssociation   O   1.3.6.1.1.6.1.8
      pcimRuleValidityAssociation   O   1.3.6.1.1.6.1.9
      pcimRuleActionAssociation   O   1.3.6.1.1.6.1.10
      pcimConditionAuxClass   O   1.3.6.1.1.6.1.11
      pcimTPCAuxClass   O   1.3.6.1.1.6.1.12
      pcimConditionVendorAuxClass   O   1.3.6.1.1.6.1.13
      pcimActionAuxClass   O   1.3.6.1.1.6.1.14
      pcimActionVendorAuxClass   O   1.3.6.1.1.6.1.15
      pcimPolicyInstance   O   1.3.6.1.1.6.1.16
      pcimElementAuxClass   O   1.3.6.1.1.6.1.17
      pcimRepository   O   1.3.6.1.1.6.1.18
      pcimRepositoryAuxClass   O   1.3.6.1.1.6.1.19
      pcimRepositoryInstance   O   1.3.6.1.1.6.1.20
      pcimSubtreesPtrAuxClass   O   1.3.6.1.1.6.1.21
      pcimGroupContainmentAuxClass   O   1.3.6.1.1.6.1.22
      pcimRuleContainmentAuxClass   O   1.3.6.1.1.6.1.23
      pcimKeywords   A   1.3.6.1.1.6.2.3
      pcimGroupName   A   1.3.6.1.1.6.2.4
      pcimRuleName   A   1.3.6.1.1.6.2.5
      pcimRuleEnabled   A   1.3.6.1.1.6.2.6
      pcimRuleConditionListType   A   1.3.6.1.1.6.2.7
      pcimRuleConditionList   A   1.3.6.1.1.6.2.8
      pcimRuleActionList   A   1.3.6.1.1.6.2.9
      pcimRuleValidityPeriodList   A   1.3.6.1.1.6.2.10
      pcimRuleUsage   A   1.3.6.1.1.6.2.11
      pcimRulePriority   A   1.3.6.1.1.6.2.12
      pcimRuleMandatory   A   1.3.6.1.1.6.2.13
      pcimRuleSequencedActions   A   1.3.6.1.1.6.2.14
      pcimRoles   A   1.3.6.1.1.6.2.15
      pcimConditionGroupNumber   A   1.3.6.1.1.6.2.16

```
   NAME                            Type    OID
   ---------------                 ----    ------------
   pcimConditionNegated            A       1.3.6.1.1.6.2.17
   pcimConditionName               A       1.3.6.1.1.6.2.18
   pcimConditionDN                 A       1.3.6.1.1.6.2.19
   pcimValidityConditionName       A       1.3.6.1.1.6.2.20
   pcimTimePeriodConditionDN       A       1.3.6.1.1.6.2.21
   pcimActionName                  A       1.3.6.1.1.6.2.22
   pcimActionOrder                 A       1.3.6.1.1.6.2.23
   pcimActionDN                    A       1.3.6.1.1.6.2.24
   pcimTPCTime                     A       1.3.6.1.1.6.2.25
   pcimTPCMonthOfYearMask          A       1.3.6.1.1.6.2.26
   pcimTPCDayOfMonthMask           A       1.3.6.1.1.6.2.27
   pcimTPCDayOfWeekMask            A       1.3.6.1.1.6.2.28
   pcimTPCTimeOfDayMask            A       1.3.6.1.1.6.2.29
   pcimTPCLocalOrUtcTime           A       1.3.6.1.1.6.2.30
   pcimVendorConstraintData        A       1.3.6.1.1.6.2.31
   pcimVendorConstraintEncoding    A       1.3.6.1.1.6.2.32
   pcimVendorActionData            A       1.3.6.1.1.6.2.33
   pcimVendorActionEncoding        A       1.3.6.1.1.6.2.34
   pcimPolicyInstanceName          A       1.3.6.1.1.6.2.35
   pcimRepositoryName              A       1.3.6.1.1.6.2.36
   pcimSubtreesAuxContainedSet     A       1.3.6.1.1.6.2.37
   pcimGroupsAuxContainedSet       A       1.3.6.1.1.6.2.38
   pcimRulesAuxContainedSet        A       1.3.6.1.1.6.2.39
```

   where Type A is Attribute, Type O is ObjectClass

   These assignments are recorded in the following registry:

      http://www.iana.org/assignments/ldap-parameters

9. Acknowledgments

   We would like to thank Kurt Zeilenga, Roland Hedburg, and Steven Legg
   for doing a review of this document and making many helpful
   suggestions and corrections.

   Several of the policy classes in this model first appeared in early
   IETF drafts on IPsec policy and QoS policy. The authors of these
   drafts were Partha Bhattacharya, Rob Adams, William Dixon, Roy
   Pereira, Raju Rajan, Jean-Christophe Martin, Sanjay Kamat, Michael
   See, Rajiv Chaudhury, Dinesh Verma, George Powers, and Raj Yavatkar.

   This document is closely aligned with the work being done in the
   Distributed Management Task Force (DMTF) Policy and Networks working
   groups. We would especially like to thank Lee Rafalow, Glenn Waters,
   David Black, Michael Richardson, Mark Stevens, David Jones, Hugh
   Mahon, Yoram Snir, and Yoram Ramberg for their helpful comments.

10. Appendix: Constructing the Value of orderedCIMKeys

   This appendix is non-normative, and is included in this document as a
   guide to implementers that wish to exchange information between CIM
   schemata and LDAP schemata.

   Within a CIM name space, the naming is basically flat; all instances
   are identified by the values of their key properties, and each
   combination of key values must be unique. A limited form of
   hierarchical naming is available in CIM, however, by using weak
   associations: since a weak association involves propagation of key
   properties and their values from the superior object to the
   subordinate one, the subordinate object can be thought of as being
   named "under" the superior object. Once they have been propagated,
   however, propagated key properties and their values function in
   exactly the same way that native key properties and their values do
   in identifying a CIM instance.

   The CIM mapping document [6] introduces a special attribute,
   orderedCIMKeys, to help map from the CIM_ManagedElement class to the
   LDAP class dlm1ManagedElement. This attribute SHOULD only be used in
   an environment where it is necessary to map between an
   LDAP-accessible directory and a CIM repository. For an LDAP
   environment, other LDAP naming attributes are defined (i.e., cn and a
   class-specific naming attribute) that SHOULD be used instead.

   The role of orderedCIMKeys is to represent the information necessary
   to correlate an entry in an LDAP-accessible directory with an
   instance in a CIM name space. Depending on how naming of CIM-related
   entries is handled in an LDAP directory, the value of orderedCIMKeys
   represents one of two things:

     - If the DIT hierarchy does not mirror the "weakness hierarchy" of
       the CIM name space, then orderedCIMKeys represents all the
       keys of the CIM instance, both native and propagated.
     - If the DIT hierarchy does mirror the "weakness hierarchy" of the
       CIM name space, then orderedCIMKeys may represent either all the
       keys of the instance, or only the native keys.

   Regardless of which of these alternatives is taken, the syntax of
   orderedCIMKeys is the same - a DirectoryString of the form

       <className>.<key>=<value>[,<key>=<value>]*

   where the <key>=<value> elements are ordered by the names of the key
   properties, according to the collating sequence for US ASCII. The
   only spaces allowed in the DirectoryString are those that fall within
   a <value> element. As with alphabetizing the key properties, the

goal of suppressing the spaces is once again to make the results of string operations predictable.

The values of the <value> elements are derived from the various CIM syntaxes according to a grammar specified in [5].

11. References

11.1. Normative References

[1]   Moore, B., Ellesson,E., Strassner, J. and A. Westerinen "Policy
        Core Information Model -- Version 1 Specification", RFC 3060,
        February 2001.

[2]   Hodges, J. and R. Morgan, "Lightweight Directory Access
        Protocol (v3): Technical Specification", RFC 3377, September
        2002.

[3]   Wahl, M., Coulbeck, A., Howes,T. and S. Kille, "Lightweight
        Directory Access Protocol (v3): Attribute Syntax Definitions",
        RFC 2252, December 1997.

[4]   The Directory: Models. ITU-T Recommendation X.501, 2001.

[5]   Distributed Management Task Force, Inc., "Common Information
        Model (CIM) Specification", Version 2.2, June 14, 1999. This
        document is available on the following DMTF web page:
        http://www.dmtf.org/standards/documents/CIM/DSP0004.pdf

[6]   Distributed Management Task Force, Inc., "DMTF LDAP Schema for
        the CIM v2.5 Core Information Model", April 15, 2002. This
        document is available on the following DMTF web page:
        http://www.dmtf.org/standards/documents/DEN/DSP0123.pdf

[7]   Wahl, M., "A Summary of the X.500(96) User Schema for use with
        LDAPv3", RFC 2256, December 1997.

[8]   The Directory: Selected Attribute Types. ITU-T Recommendation
        X.520, 2001.

[9]   Zeilenga, K., Ed., "Lightweight Directory Access Protocol
        (LDAP): Additional Matching Rules", RFC 3698, February 2004.

[10]   Bradner, S., "Key words for use in RFCs to Indicate Requirement
        Levels", BCP 14, RFC 2119, March 1997.

11.2. Informative References

   [11] Hovey, R. and S. Bradner, "The Organizations Involved in the
         IETF Standards Process", BCP 11, RFC 2028, October 1996.

   [12] Strassner, J., policy architecture BOF presentation, 42nd IETF
         Meeting, Chicago, Illinois, October 1998. Minutes of this BOF
         are available at the following location:
         http://www.ietf.org/proceedings/98aug/index.html.

   [13] Yavatkar, R., Guerin, R. and D. Pendarakis, "A Framework for
         Policy-based Admission Control", RFC 2753, January 2000.

   [14] Wahl, M., Alvestrand, H., Hodges, J. and R. Morgan,
         "Authentication Methods for LDAP", RFC 2829, May 2000

   [15] Hodges, J., Morgan, R. and M. Wahl, "Lightweight Directory
         Access Protocol (v3): Extension for Transport Layer Security",
         RFC 2830, May 2000.

   [16] Zeilenga, K., "Internet Assigned Numbers Authority (IANA)
         Considerations for the Lightweight Directory Access Protocol
         (LDAP)", BCP 64, RFC 3383, September 2002.

12. Authors' Addresses

   John Strassner
   Intelliden Corporation
   90 South Cascade Avenue
   Colorado Springs, CO 80903

   Phone: +1.719.785.0648
   Fax: +1.719.785.0644
   EMail: john.strassner@intelliden.com

   Bob Moore
   IBM Corporation
   P. O. Box 12195, BRQA/B501/G206
   3039 Cornwallis Rd.
   Research Triangle Park, NC 27709-2195

   Phone: +1 919-254-4436
   Fax: +1 919-254-6243
   EMail: remoore@us.ibm.com

   Ryan Moats
   Lemur Networks, Inc.
   15621 Drexel Circle
   Omaha, NE 68135

   Phone: +1-402-894-9456
   EMail: rmoats@lemurnetworks.net

   Ed Ellesson
   3026 Carriage Trail
   Hillsborough, NC 27278

   Phone: +1 919-644-3977
   EMail: ellesson@mindspring.com

13. Full Copyright Statement

Intellectual Property

   The IETF takes no position regarding the validity or scope of any
   Intellectual Property Rights or other rights that might be claimed
   to pertain to the implementation or use of the technology
   described in this document or the extent to which any license
   under such rights might or might not be available; nor does it
   represent that it has made any independent effort to identify any
   such rights. Information on the procedures with respect to
   rights in RFC documents can be found in BCP 78 and BCP 79.

   Copies of IPR disclosures made to the IETF Secretariat and any
   assurances of licenses to be made available, or the result of an
   attempt made to obtain a general license or permission for the use
   of such proprietary rights by implementers or users of this
   specification can be obtained from the IETF on-line IPR repository
   at http://www.ietf.org/ipr.

   The IETF invites any interested party to bring to its attention
   any copyrights, patents or patent applications, or other
   proprietary rights that may cover technology that may be required
   to implement this standard. Please address the information to the
   IETF at ietf-ipr@ietf.org.

Acknowledgement