           Policy Core Information Model -- Version 1 Specification

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 presents the object-oriented information model for
   representing policy information developed jointly in the IETF Policy
   Framework WG and as extensions to the Common Information Model (CIM)
   activity in the Distributed Management Task Force (DMTF). This model
   defines two hierarchies of object classes: structural classes
   representing policy information and control of policies, and
   association classes that indicate how instances of the structural
   classes are related to each other. Subsequent documents will define
   mappings of this information model to various concrete
   implementations, for example, to a directory that uses LDAPv3 as its
   access protocol.

Table of Contents

1. Introduction

   This document presents the object-oriented information model for
   representing policy information currently under joint development in
   the IETF Policy Framework WG and as extensions to the Common
   Information Model (CIM) activity in the Distributed Management Task
   Force (DMTF). This model defines two hierarchies of object classes:
   structural classes representing policy information and control of
   policies, and association classes that indicate how instances of the
   structural classes are related to each other. Subsequent documents
   will define mappings of this information model to various concrete
   implementations, for example, to a directory that uses LDAPv3 as its
   access protocol. The components of the CIM schema are available via
   the following URL: http://www.dmtf.org/spec/cims.html [1].

   The policy classes and associations defined in this model are
   sufficiently generic to allow them to represent policies related to
   anything. However, it is expected that their initial application in
   the IETF will be for representing policies related to QoS (DiffServ
   and IntServ) and to IPSec. Policy models for application-specific
   areas such as these may extend the Core Model in several ways. The
   preferred way is to use the PolicyGroup, PolicyRule, and
   PolicyTimePeriodCondition classes directly, as a foundation for
   representing and communicating policy information. Then, specific
   subclasses derived from PolicyCondition and PolicyAction can capture
   application-specific definitions of conditions and actions of
   policies.

   Two subclasses, VendorPolicyCondition and VendorPolicyAction, are
   also included in this document, to provide a standard extension
   mechanism for vendor-specific extensions to the Policy Core
   Information Model.

   This document fits into the overall framework for representing,
   deploying, and managing policies being developed by the Policy
   Framework Working Group. It traces its origins to work that was
   originally done for the Directory-enabled Networks (DEN)
   specification, reference [5]. Work on the DEN specification by the
   DEN Ad-Hoc Working Group itself has been completed. Further work to
   standardize the models contained in it will be the responsibility of
   selected working groups of the CIM effort in the Distributed
   Management Task Force (DMTF). DMTF standardization of the core
   policy model is the responsibility of the SLA Policy working group in
   the DMTF.

This document is organized in the following manner:

o Section 2 provides a general overview of policies and how they are
   modeled.

o Section 3 presents a high-level overview of the classes and
   associations comprising the Policy Core Information Model.

o The remainder of the document presents the detailed specifications
   for each of the classes and associations.

o Appendix A overviews naming for native CIM implementations. Other
   mappings, such as LDAPv3, will have their own naming mechanisms.

o Appendix B reproduces the DMTF's Core Policy MOF specification.

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, reference
[3].

2. Modeling Policies

The classes comprising the Policy Core Information Model are intended
to serve as an extensible class hierarchy (through specialization)
for defining policy objects that enable application developers,
network administrators, and policy administrators to represent
policies of different types.

One way to think of a policy-controlled network is to first model the
network as a state machine and then use policy to control which state
a policy-controlled device should be in or is allowed to be in at any
given time. Given this approach, policy is applied using a set of
policy rules. Each policy rule consists of a set of conditions and a
set of actions. Policy rules may be aggregated into policy groups.
These groups may be nested, to represent a hierarchy of policies.

The set of conditions associated with a policy rule specifies when
the policy rule is applicable. The set of conditions can be
expressed as either an ORed set of ANDed sets of condition statements
or an ANDed set of ORed sets of statements. Individual condition
statements can also be negated. These combinations are termed,
respectively, Disjunctive Normal Form (DNF) and Conjunctive Normal
Form (CNF) for the conditions.

If the set of conditions associated with a policy rule evaluates to
TRUE, then a set of actions that either maintain the current state of
the object or transition the object to a new state may be executed.

For the set of actions associated with a policy rule, it is possible
to specify an order of execution, as well as an indication of whether
the order is required or merely recommended. It is also possible to
indicate that the order in which the actions are executed does not
matter.

Policy rules themselves can be prioritized. One common reason for
doing this is to express an overall policy that has a general case
with a few specific exceptions.

For example, a general QoS policy rule might specify that traffic
originating from members of the engineering group is to get Bronze
Service. A second policy rule might express an exception: traffic
originating from John, a specific member of the engineering group, is
to get Gold Service. Since traffic originating from John satisfies
the conditions of both policy rules, and since the actions associated
with the two rules are incompatible, a priority needs to be
established. By giving the second rule (the exception) a higher
priority than the first rule (the general case), a policy
administrator can get the desired effect: traffic originating from
John gets Gold Service, and traffic originating from all the other
members of the engineering group gets Bronze Service.

Policies can either be used in a stand-alone fashion or aggregated
into policy groups to perform more elaborate functions. Stand-alone
policies are called policy rules. Policy groups are aggregations of
policy rules, or aggregations of policy groups, but not both. Policy
groups can model intricate interactions between objects that have
complex interdependencies. Examples of this include a sophisticated
user logon policy that sets up application access, security, and
reconfigures network connections based on a combination of user
identity, network location, logon method and time of day. A policy
group represents a unit of reusability and manageability in that its
management is handled by an identifiable group of administrators and
its policy rules would be consistently applied

Stand-alone policies are those that can be expressed in a simple
statement. They can be represented effectively in schemata or MIBs.
Examples of this are VLAN assignments, simple YES/NO QoS requests,
and IP address allocations. A specific design goal of this model is
to support both stand-alone and aggregated policies.

Policy groups and rules can be classified by their purpose and
intent. This classification is useful in querying or grouping policy
rules. It indicates whether the policy is used to motivate when or
how an action occurs, or to characterize services (that can then be
used, for example, to bind clients to network services). Describing
each of these concepts in more detail,

   o Motivational Policies are solely targeted at whether or how a
      policy's goal is accomplished. Configuration and Usage Policies
      are specific kinds of Motivational Policies. Another example is
      the scheduling of file backup based on disk write activity from
      8am to 3pm, M-F.

   o Configuration Policies define the default (or generic) setup of a
      managed entity (for example, a network service). Examples of
      Configuration Policies are the setup of a network forwarding
      service or a network-hosted print queue.

   o Installation Policies define what can and cannot be put on a
      system or component, as well as the configuration of the
      mechanisms that perform the install. Installation policies
      typically represent specific administrative permissions, and can
      also represent dependencies between different components (e.g., to
      complete the installation of component A, components B and C must
      be previously successfully installed or uninstalled).

   o Error and Event Policies. For example, if a device fails between
      8am and 9pm, call the system administrator, otherwise call the
      Help Desk.

   o Usage Policies control the selection and configuration of entities
      based on specific "usage" data. Configuration Policies can be
      modified or simply re-applied by Usage Policies. Examples of
      Usage Policies include upgrading network forwarding services after
      a user is verified to be a member of a "gold" service group, or
      reconfiguring a printer to be able to handle the next job in its
      queue.

   o Security Policies deal with verifying that the client is actually
      who the client purports to be, permitting or denying access to
      resources, selecting and applying appropriate authentication
      mechanisms, and performing accounting and auditing of resources.

   o Service Policies characterize network and other services (not use
      them). For example, all wide-area backbone interfaces shall use a
      specific type of queuing.

      Service policies describe services available in the network.
      Usage policies describe the particular binding of a client of the
      network to services available in the network.

   These categories are represented in the Policy Core Information Model
   by special values defined for the PolicyKeywords property of the
   abstract class Policy.

2.1. Policy Scope

   Policies represent business goals and objectives. A translation must
   be made between these goals and objectives and their realization in
   the network. An example of this could be a Service Level Agreement
   (SLA), and its objectives and metrics (Service Level Objectives, or
   SLOs), that are used to specify services that the network will
   provide for a given client. The SLA will usually be written in
   high-level business terminology. SLOs address more specific metrics
   in support of the SLA. These high-level descriptions of network
   services and metrics must be translated into lower-level, but also
   vendor-and device-independent specifications. The Policy Core
   Information Model classes are intended to serve as the foundation for
   these lower-level, vendor- and device-independent specifications.

   It is envisioned that the definition of the Policy Core Informational
   Model in this document is generic in nature and is applicable to
   Quality of Service (QoS), to non-QoS networking applications (e.g.,
   DHCP and IPSec), and to non-networking applications (e.g., backup
   policies, auditing access, etc.).

2.2. Declarative versus Procedural Model

   The design of the Policy Core Information Model is influenced by a
   declarative, not procedural, approach. More formally, a declarative
   language is used to describe relational and functional languages.
   Declarative languages describe relationships between variables in
   terms of functions or inference rules, to which the interpreter or
   compiler can apply a fixed algorithm in order to produce a result.
   An imperative (or procedural) language specifies an explicit sequence
   of steps to follow in order to produce a result.

   It is important to note that this information model does not rule out
   the use of procedural languages. Rather, it recognizes that both
   declarative as well as procedural languages can be used to implement
   policy. This information model is better viewed as being declarative
   because the sequence of steps for doing the processing of declarative
   statements tends to be left to the implementer. However, we have
   provided the option of expressing the desired order of action
   execution in this policy information model, and for expressing
   whether the order is mandatory or not. In addition, rather than
   trying to define algorithms or sets of instructions or steps that
   must be followed by a policy rule, we instead define a set of modular
   building blocks and relationships that can be used in a declarative
   or procedural fashion to define policies.

Compare this to a strictly procedural model. Taking such an approach
would require that we specify the condition testing sequence, and the
action execution sequence, in the policy repository itself. This
would, indeed, constrain the implementer. This is why the policy
model is characterized as a declarative one. That is, the
information model defines a set of attributes, and a set of entities
that contain these attributes. However, it does NOT define either
the algorithm to produce a result using the attributes or an explicit
sequence of steps to produce a result.

There are several design considerations and trade-offs to make in
this respect.

1. On the one hand, we would like a policy definition language to be
   reasonably human-friendly for ease of definitions and diagnostics.
   On the other hand, given the diversity of devices (in terms of
   their processing capabilities) which could act as policy decision
   points, we would like to keep the language somewhat machine-
   friendly. That is, it should be relatively simple to automate the
   parsing and processing of the language in network elements. The
   approach taken is to provide a set of classes and attributes that
   can be combined in either a declarative or procedural approach to
   express policies that manage network elements and services. The
   key point is to avoid trying to standardize rules or sets of steps
   to be followed in defining a policy. These must be left up to an
   implementation. Interoperability is achieved by standardizing the
   building blocks that are used to represent policy data and
   information.

2. An important decision to make is the semantic style of the
   representation of the information.

   The declarative approach that we are describing falls short of
   being a "true" declarative model. Such a model would also specify
   the algorithms used to combine the information and policy rules to
   achieve particular behavior. We avoid specifying algorithms for
   the same reason that we avoid specifying sets of steps to be
   followed in a policy rule. However, the design of the information
   model more closely follows that of a declarative language, and may
   be easier to understand if such a conceptual model is used. This
   leads to our third point, acknowledging a lack of "completeness"
   and instead relying on presenting information that the policy
   processing entity will work with.

3. It is important to control the complexity of the specification,
   trading off richness of expression of data in the core information
   model for ease of implementation and use. It is important to
   acknowledge the collective lack of experience in the field

regarding policies to control and manage network services and
hence avoid the temptation of aiming for "completeness". We
should instead strive to facilitate definition of a set of common
policies that customers require today (e.g., VPN and QoS) and
allow migration paths towards supporting complex policies as
customer needs and our understanding of these policies evolve with
experience. Specifically, in the context of the declarative style
language discussed above, it is important to avoid having full
blown predicate calculus as the language, as it would render many
important problems such as consistency checking and policy
decision point algorithms intractable. It is useful to consider a
reasonably constrained language from these perspectives.

The Policy Core Information Model strikes a balance between
complexity and lack of power by using the well understood logical
concepts of Disjunctive Normal Form and Conjunctive Normal Form for
combining simple policy conditions into more complex ones.

3. Overview of the Policy Core Information Model

The following diagram provides an overview of the five central
classes comprising the Policy Core Information Model, their
associations to each other, and their associations to other classes
in the overall CIM model. Note that the abstract class Policy and
the two extension classes VendorPolicyCondition and
VendorPolicyAction are not shown.

NOTE: For cardinalities, "*" is an abbreviation for "0..n".

```
                             +-----------+
                             |  System   |
           .....             +--^-----^--+              .....
           .   .                1.    1.                .   .
         *.(a).*                 .(b)  .(c)           *.(d).*
      +--v---v---------+          .     .          +-v---v------------+
      |  PolicyGroup   <........        .          | PolicyRepository |
      |                | w *           .          |                  |
      +------^---------+                .          +-----^---------^--+
           *.                           .          0..1 .    0..1 .
            .(e)                        .                .(f)       .(g)
           *.                           .                .          .
      +------v------+ w *               .                .          .
      |            <.................                .          .
      | PolicyRule |                                    .          .
      |            |                                    .          .
      |            |                                    .          .
      |            <........................            .          .
      |            |*        (h)                        .          .
      |            |                                    .          .
      |            |                                    .          .
      |            |                                    .          .
      |            |                                    .          .
      |            |                                    .          .
      |            |                                  .*        .*           .
      |            |                       +---------v-------v--+          .
      |            |                       |  PolicyCondition   |          .
      |            |                      *+-------------------+          .
      |            |         (i)                       ^               .
      |            <.............            I               .
      |            |*           .            I               .
      |            |          .*             ^               .
      |            |       +----v--------------------+          .
      |            |       | PolicyTimePeriodCondition |          .
      |            |       +-------------------------+          .
      |            |         (j)                                 .
      |            <........................            .
      |            |*                      .                     .
      |            |                     .*                      .
      |            |       +----------v---------+*          .
      |            |       |  PolicyAction      <.......
      +-----------+        +-------------------+
```
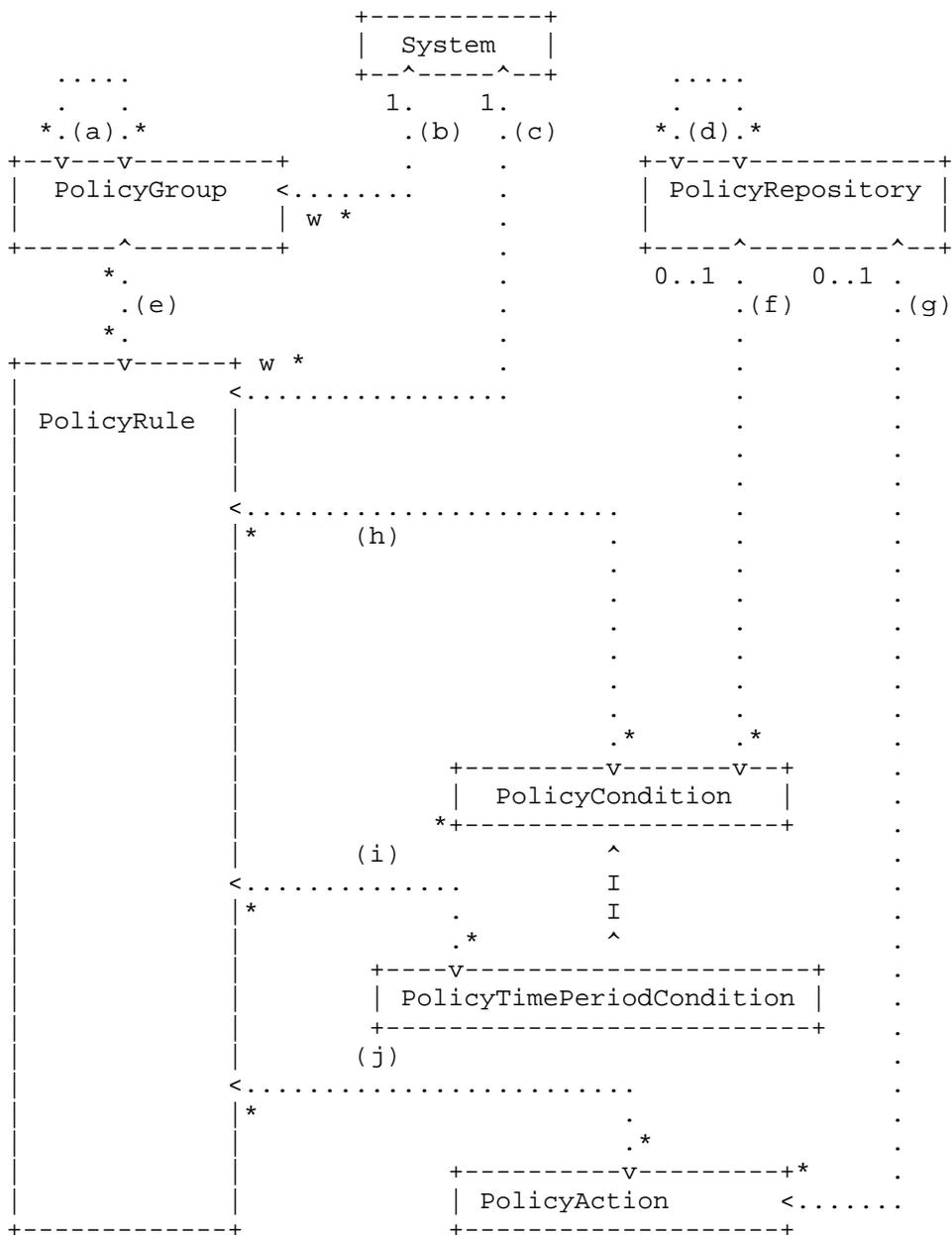
Figure 1.   Overview of the Core Policy Classes and Relationships

In this figure the boxes represent the classes, and the dotted arrows
represent the associations. The following associations appear:

(a)   PolicyGroupInPolicyGroup

(b)   PolicyGroupInSystem

(c)   PolicyRuleInSystem

(d)   PolicyRepositoryInPolicyRepository

(e)   PolicyRuleInPolicyGroup

(f)   PolicyConditionInPolicyRepository

(g)   PolicyActionInPolicyRepository

(h)   PolicyConditionInPolicyRule

(i)   PolicyRuleValidityPeriod

(j)   PolicyActionInPolicyRule

An association always connects two classes. The "two" classes may,
however, be the same class, as is the case with the
PolicyGroupInPolicyGroup association, which represents the recursive
containment of PolicyGroups in other PolicyGroups. The
PolicyRepositoryInPolicyRepository association is recursive in the
same way.

An association includes cardinalities for each of the related
classes. These cardinalities indicate how many instances of each
class may be related to an instance of the other class. For example,
the PolicyRuleInPolicyGroup association has the cardinality range "*'
(that is, "0..n") for both the PolicyGroup and PolicyRule classes.
These ranges are interpreted as follows:

o The "*" written next to PolicyGroup indicates that a PolicyRule
   may be related to no PolicyGroups, to one PolicyGroup, or to more
   than one PolicyGroup via the PolicyRuleInPolicyGroup association.
   In other words, a PolicyRule may be contained in no PolicyGroups,
   in one PolicyGroups, or in more than one PolicyGroup.

o The "*" written next to PolicyRule indicates that a PolicyGroup
   may be related to no PolicyRules, to one PolicyRule, or to more
   than one PolicyRule via the PolicyRuleInPolicyGroup association.
   In other words, a PolicyGroup may contain no PolicyRules, one
   PolicyRule, or more than one PolicyRule.

   The "w" written next to the PolicyGroupInSystem and
   PolicyRuleInSystem indicates that these are what CIM terms
   "aggregations with weak references", or more briefly, "weak
   aggregations". A weak aggregation is simply an indication of a
   naming scope. Thus these two aggregations indicate that an instance
   of a PolicyGroup or PolicyRule is named within the scope of a System
   object. A weak aggregation implicitly has the cardinality 1..1 at
   the end opposite the 'w'.

   The associations shown in Figure 1 are discussed in more detail in
   Section 7.

4. Inheritance Hierarchies for the Policy Core Information Model

   The following diagram illustrates the inheritance hierarchy for the
   core policy classes:

```
    ManagedElement (abstract)
     |
     +--Policy (abstract)
     |   |
     |   +---PolicyGroup
     |   |
     |   +---PolicyRule
     |   |
     |   +---PolicyCondition (abstract)
     |   |           |
     |   |           +---PolicyTimePeriodCondition
     |   |           |
     |   |           +---VendorPolicyCondition
     |   |
     |   +---PolicyAction (abstract)
     |               |
     |               +---VendorPolicyAction
     |
     +--ManagedSystemElement (abstract)
         |
         +--LogicalElement (abstract)
            |
            +--System (abstract)
               |
               +--AdminDomain (abstract)
                  |
                  +---PolicyRepository
```
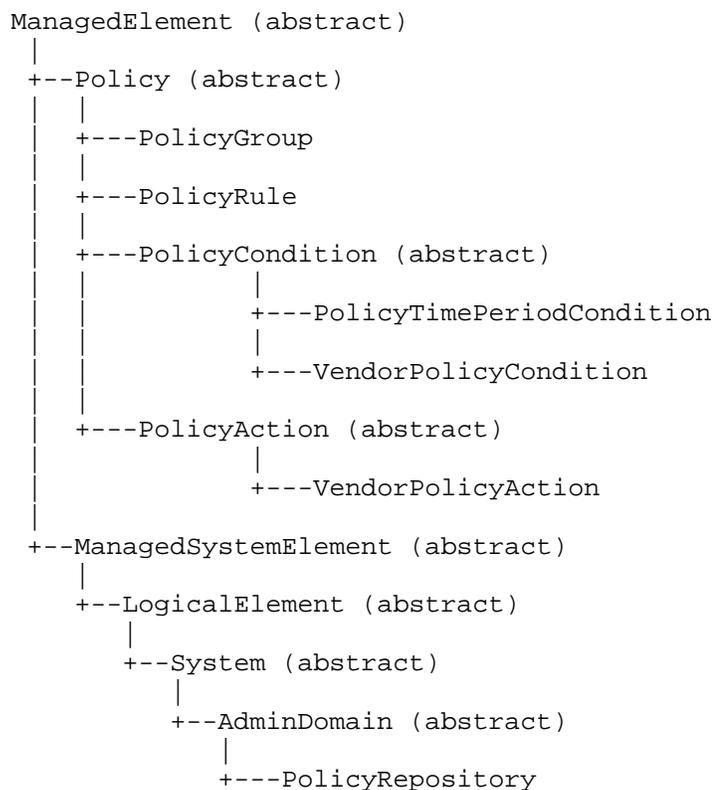
   Figure 2.   Inheritance Hierarchy for the Core Policy Classes

ManagedElement, ManagedSystemElement, LogicalElement, System, and
AdminDomain are defined in the CIM schema [1]. These classes are not
discussed in detail in this document.

In CIM, associations are also modeled as classes. For the Policy
Core Information Model, the inheritance hierarchy for the
associations is as follows:

```
    [unrooted]
      |
     +---PolicyComponent (abstract)
      |    |
      |    +---PolicyGroupInPolicyGroup
      |    |
      |    +---PolicyRuleInPolicyGroup
      |    |
      |    +---PolicyConditionInPolicyRule
      |    |
      |    +---PolicyRuleValidityPeriod
      |    |
      |    +---PolicyActionInPolicyRule
      |
     +---Dependency (abstract)
      |    |
      |    +---PolicyInSystem (abstract)
      |         |
      |         +---PolicyGroupInSystem
      |         |
      |         +---PolicyRuleInSystem
      |         |
      |         +---PolicyConditionInPolicyRepository
      |         |
      |         +---PolicyActionInPolicyRepository
      |
     +---Component (abstract)
           |
          +---SystemComponent
               |
               +---PolicyRepositoryInPolicyRepository
```

Figure 3.   Inheritance Hierarchy for the Core Policy Associations

The Dependency, Component, and SystemComponent associations are
defined in the CIM schema [1], and are not discussed further in this
document.

4.1. Implications of CIM Inheritance

   From the CIM schema, both properties and associations are inherited
   to the Policy classes. For example, the class ManagedElement is
   referenced in the associations Dependency, Statistics and
   MemberOfCollection. And, the Dependency association is in turn
   referenced in the DependencyContext association. At this very
   abstract and high level in the inheritance hierarchy, the number of
   these associations is very small and their semantics are quite
   general.

   Many of these inherited associations convey additional semantics that
   are not needed in understanding the Policy Core Information Model.
   In fact, they are defined as OPTIONAL in the CIM Schema - since their
   cardinality is "0..n" on all references. The PCIM document
   specifically discusses what is necessary to support and instantiate.
   For example, through subclassing of the Dependency association, the
   exact Dependency semantics in PCIM are described.

   So, one may wonder what to do with these other inherited
   associations. The answer is "ignore them unless you need them". You
   would need them to describe additional information and semantics for
   policy data. For example, it may be necessary to capture statistical
   data for a PolicyRule (either for the rule in a repository or for
   when it is executing in a policy system). Some examples of
   statistical data for a rule are the number of times it was
   downloaded, the number of times its conditions were evaluated, and
   the number of times its actions were executed. (These types of data
   would be described in a subclass of CIM_StatisticalInformation.) In
   these cases, the Statistics association inherited from ManagedElement
   to PolicyRule may be used to describe the tie between an instance of
   a PolicyRule and the set of statistics for it.

5. Details of the Model

   The following subsections discuss several specific issues related to
   the Policy Core Information Model.

5.1. Reusable versus Rule-Specific Conditions and Actions

   Policy conditions and policy actions can be partitioned into two
   groups: ones associated with a single policy rule, and ones that are
   reusable, in the sense that they may be associated with more than one
   policy rule. Conditions and actions in the first group are termed
   "rule-specific" conditions and actions; those in the second group are
   characterized as "reusable".

It is important to understand that the difference between a rule-
specific condition or action and a reusable one is based on the
intent of the policy administrator for the condition or action,
rather than on the current associations in which the condition or
action participates. Thus a reusable condition or action (that is,
one that a policy administrator has created to be reusable) may at
some point in time be associated with exactly one policy rule,
without thereby becoming rule-specific.

There is no inherent difference between a rule-specific condition or
action and a reusable one. There are, however, differences in how
they are treated in a policy repository. For example, it's natural
to make the access permissions for a rule-specific condition or
action identical to those for the rule itself. It's also natural for
a rule-specific condition or action to be removed from the policy
repository at the same time the rule is. With reusable conditions
and actions, on the other hand, access permissions and existence
criteria must be expressible without reference to a policy rule.

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 model itself.

An issue is highlighted by reusable and rule-specific policy
conditions and policy actions: the lack of a programmatic capability
for expressing complex constraints involving multiple associations.
Taking PolicyCondition as an example, there are two aggregations to
look at. PolicyConditionInPolicyRule has the cardinality * at both
ends, and PolicyConditionInPolicyRepository has the cardinality * at
the PolicyCondition end, and [0..1] at the PolicyRepository end.

Globally, these cardinalities are correct. However, there's more to
the story, which only becomes clear if we examine the cardinalities
separately for the two cases of a rule-specific PolicyCondition and a
reusable one.

For a rule-specific PolicyCondition, the cardinality of
PolicyConditionInPolicyRule at the PolicyRule end is [1..1], rather
than [0..n] (recall that * is an abbreviation for [0..n]), since the
condition is unique to one policy rule. And the cardinality of
PolicyConditionInPolicyRepository at the PolicyRepository end is
[0..0], since the condition is not in the "re-usable" repository.
This is OK, since these are both subsets of the specified
cardinalities.

For a reusable PolicyCondition, however, the cardinality of
PolicyConditionInPolicyRepository at the PolicyRepository end is
[1..1], since the condition must be in the repository. And, the
cardinality of PolicyConditionInPolicyRule at the PolicyRule end is
[0..n]. This last point is important: a reusable PolicyCondition
may be associated with 0, 1, or more than 1 PolicyRules, via exactly
the same association PolicyConditionInPolicyRule that binds a rule-
specific condition to its PolicyRule.

Currently the only way to document constraints of this type is
textually. More formal methods for documenting complex constraints
are needed.

5.2. Roles

5.2.1. Roles and Role Combinations

The concept of role is central to the design of the entire Policy
Framework. The idea behind roles is a simple one. Rather than
configuring, and then later having to update the configuration of,
hundreds or thousands (or more) of resources in a network, a policy
administrator assigns each resource to one or more roles, and then
specifies the policies for each of these roles. The Policy Framework
is then responsible for configuring each of the resources associated
with a role in such a way that it behaves according to the policies
specified for that role. When network behavior must be changed, the
policy administrator can perform a single update to the policy for a
role, and the Policy Framework will ensure that the necessary
configuration updates are performed on all the resources playing that
role.

A more formal definition of a role is as follows:

    A role is a type of attribute that is used to select one or more
    policies for a set of entities and/or components from among a much
    larger set of available policies.

Roles can be combined together. Here is a formal definition of a
"role- combination":

    A role-combination is a set of attributes that are used to select
    one or more policies for a set of entities and/or components from
    among a much larger set of available policies. As the examples
    below illustrate, the selection process for a role combination
    chooses policies associated with the combination itself, policies
    associated with each of its sub-combinations, and policies
    associated with each of the individual roles in the role-
    combination.

It is important to note that a role is more than an attribute. A
role defines a particular function of an entity or component that can
be used to identify particular behavior associated with that entity
or component. This difference is critical, and is most easily
understood by thinking of a role as a selector. When used in this
manner, one role (or role-combination) selects a different set of
policies than a different role (or role-combination) does.

Roles and role-combinations are especially useful in selecting which
policies are applicable to a particular set of entities or components
when the policy repository can store thousands or hundreds of
thousands of policies. This use emphasizes the ability of the role
(or role- combination) to select the small subset of policies that
are applicable from a huge set of policies that are available.

An example will illustrate how role-combinations actually work.
Suppose an installation has three roles defined for interfaces:
"Ethernet", "Campus", and "WAN". In the Policy Repository, some
policy rules could be associated with the role "Ethernet"; these
rules would apply to all Ethernet interfaces, regardless of whether
they were on the campus side or the WAN side. Other rules could be
associated with the role-combination "Campus"+"Ethernet"; these rules
would apply to the campus-side Ethernet interfaces, but not to those
on the WAN side. Finally, a third set of rules could be associated
with the role-combination "Ethernet"+"WAN"; these rules would apply
to the WAN-side Ethernet interfaces, but not to those on the campus
side. (The roles in a role-combination appear in alphabetical order
in these examples, because that is how they appear in the information
model.)

If we have a specific interface A that's associated with the role-
combination "Ethernet"+"WAN", we see that it should have three
categories of policy rules applied to it: those for the "Ethernet"
role, those for the "WAN" role, and those for the role-combination
"Ethernet"+"WAN". Going one step further, if interface B is
associated with the role- combination "branch-
office"+"Ethernet"+"WAN", then B should have seven categories of
policy rules applied to it - those associated with the following
role-combinations:

    o "branch-office"
    o "Ethernet"
    o "WAN"
    o "branch-office"+"Ethernet"
    o "branch-office"+"WAN"
    o "Ethernet"+"WAN"
    o "branch-office"+"Ethernet"+"WAN".

In order to get all of the right policy rules for a resource like
interface B, a PDP must expand the single role-combination it
receives for B into this list of seven role-combinations, and then
retrieve from the Policy Repository the corresponding seven sets of
policy rules. Of course this example is unusually complicated: the
normal case will involve expanding a two-role combination into three
values identifying three sets of policy rules.

Role-combinations also help to simplify somewhat the problem of
identifying conflicts between policy rules. With role-combinations,
it is possible for a policy administrator to specify one set of
policy rules for campus-side Ethernet interfaces, and a second set of
policy rules for WAN-side Ethernet interfaces, without having to
worry about conflicts between the two sets of rules. The policy
administrator simply "turns off" conflict detection for these two
sets of rules, by telling the policy management system that the roles
"Campus" and "WAN" are incompatible with each other. This indicates
that the role combination will never occur, and therefore conflicts
will never occur. In some cases the technology itself might identify
incompatible roles: "Ethernet" and "FrameRelay", for example. But
for less precise terms like "Campus" and "WAN", the policy
administrator must say whether they identify incompatible roles.

When the policy administrator does this, there are three effects:

1. If an interface has assigned to it a role-combination involving
   both "Campus" and "WAN", then the policy management system can
   flag it as an error.

2. If a policy rule is associated with a role-combination involving
   both "Campus" and "WAN", then the policy management system can
   flag it as an error.

3. If the policy management system sees two policy rules, where one
   is tied to the role "Campus" (or to a role-combination that
   includes the role "Campus") and the other is tied to the role
   "WAN" (or to a role- combination that includes the role "WAN"),
   then the system does not need to look for conflicts between the
   two policy rules: because of the incompatible roles, the two
   rules cannot possibly conflict.

```
                      +-------------------+
                      | Policy Repository |
                      +-------------------+
                              V
                              V retrieval of policy
                              V
                        +---------+
                        | PDP/PEP |
                        +---------+
                              v
                              v application of policy
                              v
                      +----------------+
                      | Network Entity |
                      +----------------+
```
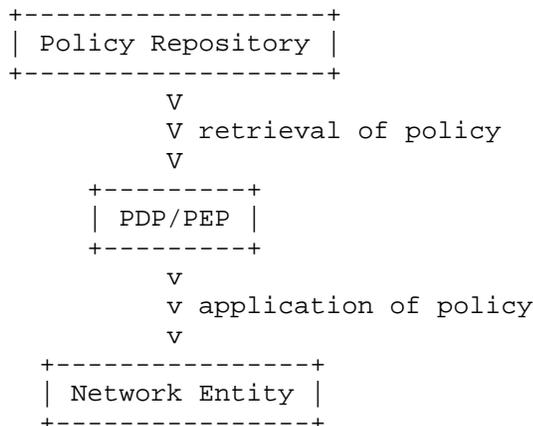
             Figure 4.   Retrieval and Application of a Policy

     Figure 4, which is introduced only as an example of how the Policy
     Framework might be implemented by a collection of network
     components, illustrates how roles operate within the Policy
     Framework. Because the distinction between them is not important
     to this discussion, the PDP and the PEP are combined in one box.
     The points illustrated here apply equally well, though, to an
     environment where the PDP and the PEP are implemented separately.

     A role represents a functional characteristic or capability of a
     resource to which policies are applied. Examples of roles include
     Backbone interface, Frame Relay interface, BGP-capable router, web
     server, firewall, etc. The multiple roles assigned to a single
     resource are combined to form that resource's role combination.
     Role combinations are represented in the PCIM by values of the
     PolicyRoles property in the PolicyRule class. A PDP uses policy
     roles as follows to identify the policies it needs to be aware of:

     1. The PDP learns in some way the list of roles that its PEPs
        play. This information might be configured at the PDP, the
        PEPs might supply it to the PDP, or the PDP might retrieve it
        from a repository.

     2. Using repository-specific means, the PDP determines where to
        look for policy rules that might apply to it.

     3. Using the roles and role-combinations it received from its PEPs
        as indicated in the examples above, the PDP is able to locate
        and retrieve the policy rules that are relevant to it.

5.2.2. The PolicyRoles Property

   As indicated earlier, PolicyRoles is a property associated with a
   policy rule. It is an array holding "role combinations" for the
   policy rule, and correlates with the roles defined for a network
   resource. Using the PolicyRoles property, it is possible to mark a
   policy rule as applying, for example, to a Frame Relay interface or
   to a backbone ATM interface. The PolicyRoles property take strings
   of the form:

      <RoleName>[&&<RoleName>]*

   Each value of this property represents a role combination, including
   the special case of a "combination" containing only one role. As the
   format indicates, the role names in a role combination are ANDed
   together to form a single selector. The multiple values of the
   PolicyRoles property are logically ORed, to make it possible for a
   policy rule to have multiple selectors.

   The individual role names in a role combination must appear in
   alphabetical order (according to the collating sequence for UCS-2
   characters), to make the string matches work correctly. The role
   names used in an environment are specified by the policy
   administrator.

5.3. Local Time and UTC Time in PolicyTimePeriodConditions

   An instance of PolicyTimePeriodCondition has up to five properties
   that represent times: TimePeriod, MonthOfYearMask, DayOfMonthMask,
   DayOfWeekMask, and TimeOfDayMask. All of the time-related properties
   in an instance of PolicyTimePeriodCondition represent one of two
   types of times: local time at the place where a policy rule is
   applied, or UTC time. The property LocalOrUtcTime indicates which
   time representation applies to an instance of
   PolicyTimePeriodCondition.

   Since the PCIM provides only for local time and UTC time, a Policy
   Management Tool that provides for other time representations (for
   example, a fixed time at a particular location) will need to map from
   these other representations to either local time or UTC time. An
   example will illustrate the nature of this mapping.

   Suppose a policy rule is tied to the hours of operation for a Help
   Desk: 0800 to 2000 Monday through Friday [US] Eastern Time. In
   order to express these times in PolicyTimePeriodCondition, a
   management tool must convert them to UTC times. (They are not local
   times, because they refer to a single time interval worldwide, not to
   intervals tied to the local clocks at the locations where the

PolicyRule is being applied.) As reference [10] points out, mapping
from [US] Eastern Time to UTC time is not simply a matter of applying
an offset: the offset between [US] Eastern Time and UTC time
switches between -0500 and -0400 depending on whether Daylight
Savings Time is in effect in the US.

Suppose the policy administrator's goal is to have a policy rule be
valid from 0800 until 1200 [US] Eastern Time on every Monday, within
the overall time period from the beginning of 2000 until the end of
2001. The Policy Management Tool could either be configured with the
definition of what [US] Eastern Time means, or it could be configured
with knowledge of where to go to get this information. Reference
[10] contains further discussion of time zone definitions and where
they might reside.

Armed with knowledge about [US] Eastern Time, the Policy Management
Tool would create however many instances of PolicyTimePeriodCondition
it needed to represent the desired intervals. Note that while there
is an increased number of PolicyTimePeriodCondition instances, there
is still just one PolicyRule, which is tied to all the
PolicyTimePeriodCondition instances via the aggregation
PolicyRuleValidityPeriod. Here are the first two of these instances:

        1. TimePeriod: 20000101T050000/20000402T070000
           DayOfWeekMask: { Monday }
           TimeOfDayMask: T130000/T170000
           LocalOrUtcTime: UTC

        2. TimePeriod: 20000402T070000/20001029T070000
           DayOfWeekMask: { Monday }
           TimeOfDayMask: T120000/T160000
           LocalOrUtcTime: UTC

There would be three more similar instances, for winter 2000-2001,
summer 2001, and winter 2001 up through December 31.

Had the example been chosen differently, there could have been even
more instances of PolicyTimePeriodCondition. If, for example, the

time interval had been from 0800 - 2200 [US] Eastern Time on Mondays,
instance 1 above would have split into two instances: one with a UTC
time interval of T130000/T240000 on Mondays, and another with a UTC
time interval of T000000/T030000 on Tuesdays. So the end result
would have been ten instances of PolicyTimePeriodCondition, not five.

By restricting PolicyTimePeriodCondition to local time and UTC time,
the PCIM places the difficult and expensive task of mapping from
"human" time representations to machine-friendly ones in the Policy

Management Tool. Another approach would have been to place in
PolicyTimePeriodCondition a means of representing a named time zone,
such as [US] Eastern Time. This, however, would have passed the
difficult mapping responsibility down to the PDPs and PEPs. It is
better to have a mapping such as the one described above done once in
a Policy Management Tool, rather than having it done over and over in
each of the PDPs (and possibly PEPs) that need to apply a PolicyRule.

5.4. CIM Data Types

   Since PCIM extends the CIM Schema, a correspondence between data
   types used in both CIM and PCIM is needed. The following CIM data
   types are used in the class definitions that follow in Sections 6 and
   7:

   o uint8   unsigned 8-bit integer

   o uint16   unsigned 16-bit integer

   o boolean   Boolean

   o string   UCS-2 string.

   Strings in CIM are stored as UCS-2 characters, where each character
   is encoded in two octets. Thus string values may need to be
   converted when moving between a CIM environment and one that uses a
   different string encoding. For example, in an LDAP-accessible
   directory, attributes of type DirectoryString are stored in UTF-8
   format. RFC 2279 [7] explains how to convert between these two
   formats.

   When it is applied to a CIM string, a MaxLen value refers to the
   maximum number of characters in the string, rather than to the
   maximum number of octets.

   In addition to the CIM data types listed above, the association
   classes in Section 7 use the following type:

   o <classname> ref   strongly typed reference.

   There is one obvious omission from this list of CIM data types:
   octet strings. This is because CIM treats octet strings as a derived
   data type. There are two forms of octet strings in CIM - an ordered
   uint8 array for single-valued strings, and a string array for multi-
   valued properties. Both are described by adding an "OctetString"
   qualifier (meta-data) to the property. This qualifier functions
   exactly like an SMIv2 (SNMP) Textual Convention, refining the syntax
   and semantics of the existing CIM data type.

The first four numeric elements of both of the "OctetString"
representations are a length field. (The reason that the "numeric"
adjective is added to the previous sentence is that the string
property also includes '0' and 'x', as its first characters.) In
both cases, these 4 numeric elements (octets) are included in
calculating the length. For example, a single-valued octet string
property having the value X'7C' would be represented by the uint8
array, X'00 00 00 05 7C'.

The strings representing the individual values of a multi-valued
property qualified with the "OctetString" qualifier are constructed
similarly:

1. Take a value to be encoded as an octet string (we'll use X'7C' as
   above), and prepend to it a four-octet length. The result is the
   same, X'00 00 00 05 7C'.

2. Convert this to a character string by introducing '0' and 'x' at
   the front, and removing all white space. Thus we have the 12-
   character string "0x000000057C". This string is the value of one
   of the array elements in the CIM string array. Since CIM uses the
   UCS-2 character set, it will require 24 octets to encode this 12-
   character string.

Mappings of the PCIM to particular data models are not required to
follow this CIM technique of representing multi-valued octet strings
as length- prefixed character strings. In an LDAP mapping, for
example, it would be much more natural to simply use the Octet String
syntax, and omit the prepended length octets.

5.5. Comparison between CIM and LDAP Class Specifications

There are a number of differences between CIM and LDAP class
specifications. The ones that are relevant to the abbreviated class
specifications in this document are listed below. These items are
included here to help introduce the IETF community, which is already
familiar with LDAP, to CIM modeling, and by extension, to information
modeling in general.

o   Instead of LDAP's three class types (abstract, auxiliary,
    structural), CIM has only two: abstract and instantiable. The
    type of a CIM class is indicated by the Boolean qualifier
    ABSTRACT.

o   CIM uses the term "property" for what LDAP terms an "attribute".

o CIM uses the array notation "[ ]" to indicate that a property is
   multi-valued. CIM defines three types of arrays: bags (contents
   are unordered, duplicates allowed), ordered bags (contents are
   ordered but duplicates are allowed) and indexed arrays (contents
   are ordered and no duplicates are allowed).

o CIM classes and properties are identified by name, not by OID.

o CIM classes use a different naming scheme for native
   implementations, than LDAP. The CIM naming scheme is documented
   in Appendix A since it is not critical to understanding the
   information model, and only applies when communicating with a
   native CIM implementation.

o In LDAP, attribute definitions are global, and the same attribute
   may appear in multiple classes. In CIM, a property is defined
   within the scope of a single class definition. The property may
   be inherited into subclasses of the class in which it is defined,
   but otherwise it cannot appear in other classes. One side effect
   of this difference is that CIM property names tend to be much
   shorter than LDAP attribute names, since they are implicitly
   scoped by the name of the class in which they are defined.

There is also a notational convention that this document follows, to
improve readability. In CIM, all class and property names are
prefixed with the characters "CIM_". These prefixes have been
omitted throughout this document, with one exception regarding
naming, documented in Appendix A.

For the complete definition of the CIM specification language, see
reference [2].

6. Class Definitions

The following sections contain the definitions of the PCIM classes.

6.1. The Abstract Class "Policy"

The abstract class Policy collects several properties that may be
included in instances of any of the Core Policy classes (or their
subclasses). For convenience, the two properties that Policy
inherits from ManagedElement in the CIM schema are shown here as
well.

The class definition is as follows:

```
NAME                Policy
DESCRIPTION         An abstract class with four properties for
                    describing a policy-related instance.
DERIVED FROM        ManagedElement
ABSTRACT            TRUE
PROPERTIES          CommonName (CN)
                    PolicyKeywords[ ]
                            // Caption (inherited)
                            // Description (inherited)
```

6.1.1. The Property "CommonName (CN)"

   The CN, or CommonName, property corresponds to the X.500 attribute
   commonName (cn). In X.500 this property specifies one or more user-
   friendly names (typically only one name) by which an object is
   commonly known, names that conform to the naming conventions of the
   country or culture with which the object is associated. In the CIM
   model, however, the CommonName property is single-valued.

```
NAME                CN
DESCRIPTION         A user-friendly name of a policy-related object.
SYNTAX              string
```

6.1.2. The Multi-valued Property "PolicyKeywords"

   This property provides a set of one or more keywords that a policy
   administrator may use to assist in characterizing or categorizing a
   policy object. Keywords are of one of two types:

   o Keywords defined in this document, or in documents that define
      subclasses of the classes defined in this document. These
      keywords provide a vendor-independent, installation-independent
      way of characterizing policy objects.

   o Installation-dependent keywords for characterizing policy objects.
      Examples include "Engineering", "Billing", and "Review in December
      2000".

   This document defines the following keywords: "UNKNOWN",
   "CONFIGURATION", "USAGE", "SECURITY", "SERVICE", "MOTIVATIONAL",
   "INSTALLATION", and "EVENT". These concepts were defined earlier in
   Section 2.

One additional keyword is defined: "POLICY". The role of this
keyword is to identify policy-related instances that would not
otherwise be identifiable as being related to policy. It may be
needed in some repository implementations.

Documents that define subclasses of the Policy Core Information Model
classes SHOULD define additional keywords to characterize instances
of these subclasses. By convention, keywords defined in conjunction
with class definitions are in uppercase. Installation-defined
keywords can be in any case.

The property definition is as follows:

```
NAME            PolicyKeywords
DESCRIPTION     A set of keywords for characterizing /categorizing
                policy objects.
SYNTAX          string
```

6.1.3. The Property "Caption" (Inherited from ManagedElement)

This property provides a one-line description of a policy-related
object.

```
NAME            Caption
DESCRIPTION     A one-line description of this policy-related object.
SYNTAX          string
```

6.1.4. The Property "Description" (Inherited from ManagedElement)

This property provides a longer description than that provided by the
caption property.

```
NAME            Description
DESCRIPTION     A long description of this policy-related object.
SYNTAX          string
```

6.2. The Class "PolicyGroup"

This class is a generalized aggregation container. It enables either
PolicyRules or PolicyGroups to be aggregated in a single container.
Loops, including the degenerate case of a PolicyGroup that contains
itself, are not allowed when PolicyGroups contain other PolicyGroups.

PolicyGroups and their nesting capabilities are shown in Figure 5
below. Note that a PolicyGroup can nest other PolicyGroups, and
there is no restriction on the depth of the nesting in sibling
PolicyGroups.

```
+-------------------------------------------------------+
|                     PolicyGroup                       |
|                                                       |
|  +-------------------+       +-----------------+      |
|  |    PolicyGroup A  |       |   PolicyGroup X |      |
|  |                   |       |                 |      |
|  |  +---------------+ |  ooo  |                 |      |
|  |  | PolicyGroup A1 | |      |                 |      |
|  |  +---------------+ |       |                 |      |
|  +-------------------+       +-----------------+      |
+-------------------------------------------------------+
```

         Figure 5.   Overview of the PolicyGroup class

As a simple example, think of the highest level PolicyGroup shown in
Figure 5 above as a logon policy for US employees of a company. This
PolicyGroup may be called USEmployeeLogonPolicy, and may aggregate
several PolicyGroups that provide specialized rules per location.
Hence, PolicyGroup A in Figure 5 above may define logon rules for
employees on the West Coast, while another PolicyGroup might define
logon rules for the Midwest (e.g., PolicyGroup X), and so forth.

Note also that the depth of each PolicyGroup does not need to be the
same. Thus, the WestCoast PolicyGroup might have several additional
layers of PolicyGroups defined for any of several reasons (different
locales, number of subnets, etc..). The PolicyRules are therefore
contained at n levels from the USEmployeeLogonPolicyGroup. Compare
this to the Midwest PolicyGroup (PolicyGroup X), which might directly
contain PolicyRules.

The class definition for PolicyGroup is as follows:

```
    NAME            PolicyGroup
    DESCRIPTION     A container for either a set of related
                    PolicyRules or a set of related PolicyGroups.
    DERIVED FROM    Policy
    ABSTRACT        FALSE
    PROPERTIES      NONE
```

No properties are defined for this class since it inherits all its
properties from Policy. The class exists to aggregate PolicyRules or
other PolicyGroups. It is directly instantiable. In an
implementation, various key/identification properties MUST be
defined. The keys for a native CIM implementation are defined in
Appendix A, Section 13.1.1. Keys for an LDAP implementation will be
defined in the LDAP mapping of this information model [11].

6.3. The Class "PolicyRule"

   This class represents the "If Condition then Action" semantics
   associated with a policy. A PolicyRule condition, in the most
   general sense, is represented as either an ORed set of ANDed
   conditions (Disjunctive Normal Form, or DNF) or an ANDed set of ORed
   conditions (Conjunctive Normal Form, or CNF). Individual conditions
   may either be negated (NOT C) or unnegated (C). The actions
   specified by a PolicyRule are to be performed if and only if the
   PolicyRule condition (whether it is represented in DNF or CNF)
   evaluates to TRUE.

   The conditions and actions associated with a policy rule are modeled,
   respectively, with subclasses of the classes PolicyCondition and
   PolicyAction. These condition and action objects are tied to
   instances of PolicyRule by the PolicyConditionInPolicyRule and
   PolicyActionInPolicyRule aggregations.

   As illustrated above in Section 3, a policy rule may also be
   associated with one or more policy time periods, indicating the
   schedule according to which the policy rule is active and inactive.
   In this case it is the PolicyRuleValidityPeriod aggregation that
   provides the linkage.

   A policy rule is illustrated conceptually in Figure 6. below.

```
       +---------------------------------------------------+
       |                    PolicyRule                     |
       |                                                   |
       | +--------------------+    +-----------------+     |
       | | PolicyCondition(s) |    | PolicyAction(s) |     |
       | +--------------------+    +-----------------+     |
       |                                                   |
       |         +-----------------------------+           |
       |         | PolicyTimePeriodCondition(s) |          |
       |         +-----------------------------+           |
       +---------------------------------------------------+
```

           Figure 6.   Overview of the PolicyRule Class

   The PolicyRule class uses the property ConditionListType, to indicate
   whether the conditions for the rule are in DNF or CNF. The
   PolicyConditionInPolicyRule aggregation contains two additional
   properties to complete the representation of the rule's conditional
   expression. The first of these properties is an integer to partition
   the referenced conditions into one or more groups, and the second is
   a Boolean to indicate whether a referenced condition is negated. An

example shows how ConditionListType and these two additional
properties provide a unique representation of a set of conditions in
either DNF or CNF.

Suppose we have a PolicyRule that aggregates five PolicyConditions C1
through C5, with the following values in the properties of the five
PolicyConditionInPolicyRule associations:

```
   C1:  GroupNumber = 1, ConditionNegated = FALSE
   C2:  GroupNumber = 1, ConditionNegated = TRUE
   C3:  GroupNumber = 1, ConditionNegated = FALSE
   C4:  GroupNumber = 2, ConditionNegated = FALSE
   C5:  GroupNumber = 2, ConditionNegated = FALSE
```

If ConditionListType = DNF, then the overall condition for the
PolicyRule is:

```
   (C1 AND (NOT C2) AND C3) OR (C4 AND C5)
```

On the other hand, if ConditionListType = CNF, then the overall
condition for the PolicyRule is:

```
   (C1 OR (NOT C2) OR C3) AND (C4 OR C5)
```

In both cases, there is an unambiguous specification of the overall
condition that is tested to determine whether to perform the actions
associated with the PolicyRule.

The class definition is as follows:

```
NAME            PolicyRule
DESCRIPTION     The central class for representing the "If Condition
                then Action" semantics associated with a policy rule.
DERIVED FROM    Policy
ABSTRACT        FALSE
PROPERTIES      Enabled
                ConditionListType
                RuleUsage
                Priority
                Mandatory
                SequencedActions
                PolicyRoles
```

The PolicyRule class is directly instantiable. In an implementation,
various key/identification properties MUST be defined. The keys for
a native CIM implementation are defined in Appendix A, Section
13.1.2. Keys for an LDAP implementation will be defined in the LDAP
mapping of this information model [11].

6.3.1. The Property "Enabled"

   This property indicates whether a policy rule is currently enabled,
   from an administrative point of view. Its purpose is to allow a
   policy administrator to enable or disable a policy rule without
   having to add it to, or remove it from, the policy repository.

   The property also supports the value 'enabledForDebug'. When the
   property has this value, the entity evaluating the policy
   condition(s) is being told to evaluate the conditions for the policy
   rule, but not to perform the actions if the conditions evaluate to
   TRUE. This value serves as a debug vehicle when attempting to
   determine what policies would execute in a particular scenario,
   without taking any actions to change state during the debugging.

   The property definition is as follows:

   NAME   Enabled
   DESCRIPTION   An enumeration indicating whether a policy rule is
                     administratively enabled, administratively disabled,
                     or enabled for debug mode.
   SYNTAX   uint16
   VALUES   enabled(1), disabled(2), enabledForDebug(3)
   DEFAULT VALUE   enabled(1)

6.3.2. The Property "ConditionListType"

   This property is used to specify whether the list of policy
   conditions associated with this policy rule is in disjunctive normal
   form (DNF) or conjunctive normal form (CNF). If this property is not
   present, the list type defaults to DNF. The property definition is
   as follows:

   NAME   ConditionListType
   DESCRIPTION   Indicates whether the list of policy conditions
                     associated with this policy rule is in disjunctive
                     normal form (DNF) or conjunctive normal form (CNF).
   SYNTAX   uint16
   VALUES   DNF(1), CNF(2)
   DEFAULT VALUE   DNF(1)

6.3.3. The Property "RuleUsage"

   This property is a free-form string that recommends how this policy
   should be used. The property definition is as follows:

    NAME   RuleUsage
    DESCRIPTION   This property is used to provide guidelines on
                        how this policy should be used.
    SYNTAX   string

6.3.4. The Property "Priority"

   This property provides a non-negative integer for prioritizing policy
   rules relative to each other. Larger integer values indicate higher
   priority. Since one purpose of this property is to allow specific,
   ad hoc policy rules to temporarily override established policy rules,
   an instance that has this property set has a higher priority than all
   instances that use or set the default value of zero.

   Prioritization among policy rules provides a basic mechanism for
   resolving policy conflicts.

   The property definition is as follows:

   NAME   Priority
   DESCRIPTION   A non-negative integer for prioritizing this
                       PolicyRule relative to other PolicyRules. A larger
                       value indicates a higher priority.
   SYNTAX   uint16
   DEFAULT VALUE   0

6.3.5. The Property "Mandatory"

   This property indicates whether evaluation (and possibly action
   execution) of a PolicyRule is mandatory or not. Its concept is
   similar to the ability to mark packets for delivery or possible
   discard, based on network traffic and device load.

   The evaluation of a PolicyRule MUST be attempted if the Mandatory
   property value is TRUE. If the Mandatory property value of a
   PolicyRule is FALSE, then the evaluation of the rule is "best effort"
   and MAY be ignored.

   The property definition is as follows:

    NAME   Mandatory
    DESCRIPTION   A flag indicating that the evaluation of the
                        PolicyConditions and execution of PolicyActions
                        (if the condition list evaluates to TRUE) is
                        required.
    SYNTAX   boolean
    DEFAULT VALUE   TRUE

6.3.6. The Property "SequencedActions"

   This property gives a policy administrator a way of specifying how
   the ordering of the policy actions associated with this PolicyRule is
   to be interpreted.   Three values are supported:

   o mandatory(1):   Do the actions in the indicated order, or don't do
      them at all.

   o recommended(2): Do the actions in the indicated order if you can,
      but if you can't do them in this order, do them in another order
      if you can.

   o dontCare(3):   Do them -- I don't care about the order.

   When error / event reporting is addressed for the Policy Framework,
   suitable codes will be defined for reporting that a set of actions
   could not be performed in an order specified as mandatory (and thus
   were not performed at all), that a set of actions could not be
   performed in a recommended order (and moreover could not be performed
   in any order), or that a set of actions could not be performed in a
   recommended order (but were performed in a different order).   The
   property definition is as follows:

```
      NAME              SequencedActions
      DESCRIPTION       An enumeration indicating how to interpret the
                        action ordering indicated via the
                        PolicyActionInPolicyRule aggregation.
      SYNTAX            uint16
      VALUES            mandatory(1), recommended(2), dontCare(3)
      DEFAULT VALUE     dontCare(3)
```

6.3.7. The Multi-valued Property "PolicyRoles"

   This property represents the roles and role combinations associated
   with a policy rule. Each value represents one role combination.
   Since this is a multi-valued property, more than one role combination
   can be associated with a single policy rule. Each value 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). The property
   definition is as follows:

```
     NAME               PolicyRoles
     DESCRIPTION        A set of strings representing the roles and role
                        combinations associated with a policy rule.  Each
                        value represents one role combination.
     SYNTAX             string
```

6.4. The Abstract Class "PolicyCondition"

   The purpose of a policy condition is to determine whether or not the
   set of actions (aggregated in the PolicyRule that the condition
   applies to) should be executed or not. For the purposes of the
   Policy Core Information Model, all that matters about an individual
   PolicyCondition is that it evaluates to TRUE or FALSE. (The
   individual PolicyConditions associated with a PolicyRule are combined
   to form a compound expression in either DNF or CNF, but this is
   accomplished via the ConditionListType property, discussed above, and
   by the properties of the PolicyConditionInPolicyRule aggregation,
   introduced above and discussed further in Section 7.6 below.) A
   logical structure within an individual PolicyCondition may also be
   introduced, but this would have to be done in a subclass of
   PolicyCondition.

   Because it is general, the PolicyCondition class does not itself
   contain any "real" conditions. These will be represented by
   properties of the domain-specific subclasses of PolicyCondition.

```
      +--------------------------------------------------------------+
      |                  Policy Conditions in DNF                    |
      | +-------------------------+     +------------------------+ |
      | |        AND list         |     |        AND list        | |
      | | +-----------------+     |     | +----------------+     | |
      | | | PolicyCondition |     |     | | PolicyCondition |    | |
      | | +-----------------+     |     | +----------------+     | |
      | | +-----------------+     |     | +----------------+     | |
      | | | PolicyCondition |     |     | | PolicyCondition |    | |
      | | +-----------------+     | ... | +----------------+     | |
      | |        ...              | ORed|        ...             | |
      | |       ANDed             |     |       ANDed            | |
      | | +-----------------+     |     | +----------------+     | |
      | | | PolicyCondition |     |     | | PolicyCondition |    | |
      | | +-----------------+     |     | +----------------+     | |
      | +-------------------------+     +------------------------+ |
      +--------------------------------------------------------------+
```
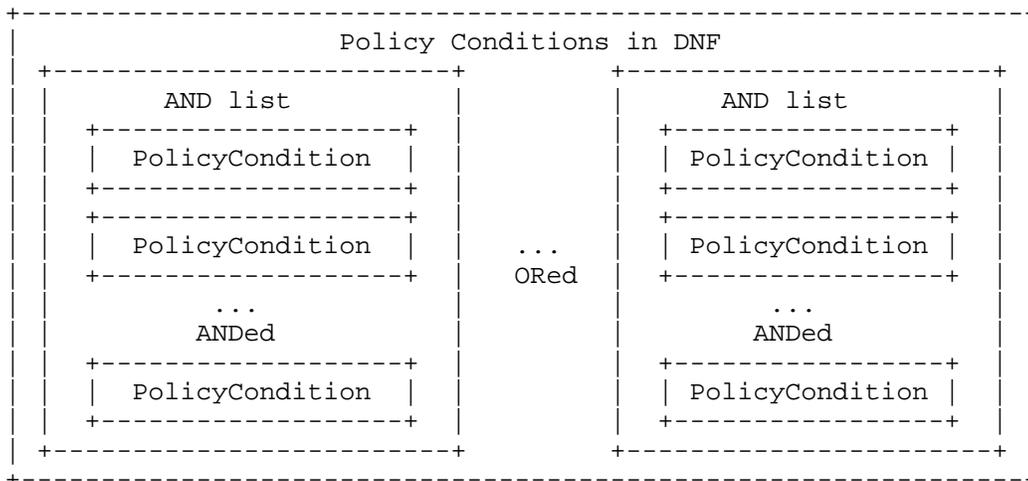
          Figure 7.   Overview of Policy Conditions in DNF

This figure illustrates that when policy conditions are in DNF, there
are one or more sets of conditions that are ANDed together to form
AND lists. An AND list evaluates to TRUE if and only if all of its
constituent conditions evaluate to TRUE. The overall condition then
evaluates to TRUE if and only if at least one of its constituent AND
lists evaluates to TRUE.

```
+--------------------------------------------------------------------+
|                     Policy Conditions in CNF                       |
| +------------------------+        +------------------------+       |
| |         OR list        |        |         OR list        |       |
| | +------------------+   |        | +----------------+     |       |
| | | PolicyCondition  |   |        | | PolicyCondition |    |       |
| | +------------------+   |        | +----------------+     |       |
| | +------------------+   |        | +----------------+     |       |
| | | PolicyCondition  |   |  ...   | | PolicyCondition |    |       |
| | +------------------+   | ANDed  | +----------------+     |       |
| |        ...             |        |        ...             |       |
| |        ORed            |        |        ORed            |       |
| | +------------------+   |        | +----------------+     |       |
| | | PolicyCondition  |   |        | | PolicyCondition |    |       |
| | +------------------+   |        | +----------------+     |       |
| +------------------------+        +------------------------+       |
+--------------------------------------------------------------------+
```

         Figure 8.   Overview of Policy Conditions in CNF

In this figure, the policy conditions are in CNF. Consequently,
there are one or more OR lists, each of which evaluates to TRUE if
and only if at least one of its constituent conditions evaluates to
TRUE. The overall condition then evaluates to TRUE if and only if
ALL of its constituent OR lists evaluate to TRUE.

The class definition of PolicyCondition is as follows:

```
NAME             PolicyCondition
DESCRIPTION      A class representing a rule-specific or reusable
                 policy condition to be evaluated in conjunction
                 with a policy rule.
DERIVED FROM     Policy
ABSTRACT         TRUE
PROPERTIES       NONE
```

No properties are defined for this class since it inherits all its
properties from Policy. The class exists as an abstract superclass
for domain-specific policy conditions, defined in subclasses. In an
implementation, various key/identification properties MUST be defined
for the class or its instantiable subclasses. The keys for a native

CIM implementation are defined in Appendix A, Section 13.2. Keys for
an LDAP implementation will be defined in the LDAP mapping of this
information model [11].

When identifying and using the PolicyCondition class, it is necessary
to remember that a condition can be rule-specific or reusable. This
was discussed above in Section 5.1. The distinction between the two
types of policy conditions lies in the associations in which an
instance can participate, and in how the different instances are
named. Conceptually, a reusable policy condition resides in a policy
repository, and is named within the scope of that repository. On the
other hand, a rule-specific policy condition is, as the name
suggests, named within the scope of the single policy rule to which
it is related.

The distinction between rule-specific and reusable PolicyConditions
affects the CIM naming, defined in Appendix A, and the LDAP mapping
[11].

6.5. The Class "PolicyTimePeriodCondition"

This class provides a means of representing the time periods during
which a policy rule is valid, i.e., active. At all times that fall
outside these time periods, the policy rule has no effect. A policy
rule is treated as valid at all times if it does not specify a
PolicyTimePeriodCondition.

In some cases a PDP may need to perform certain setup / cleanup
actions when a policy rule becomes active / inactive. For example,
sessions that were established while a policy rule was active might
need to be taken down when the rule becomes inactive. In other
cases, however, such sessions might be left up: in this case, the
effect of deactivating the policy rule would just be to prevent the
establishment of new sessions. Setup / cleanup behaviors on validity
period transitions are not currently addressed by the PCIM, and must
be specified in 'guideline' documents, or via subclasses of
PolicyRule, PolicyTimePeriodCondition or other concrete subclasses of
Policy. If such behaviors need to be under the control of the policy
administrator, then a mechanism to allow this control must also be
specified in the subclass.

PolicyTimePeriodCondition is defined as a subclass of
PolicyCondition. This is to allow the inclusion of time-based
criteria in the AND/OR condition definitions for a PolicyRule.

Instances of this class may have up to five properties identifying
time periods at different levels. The values of all the properties
present in an instance are ANDed together to determine the validity

period(s) for the instance. For example, an instance with an overall
validity range of January 1, 2000 through December 31, 2000; a month
mask that selects March and April; a day-of-the-week mask that
selects Fridays; and a time of day range of 0800 through 1600 would
represent the following time periods:

```
Friday, March  5, 2000, from 0800 through 1600;
Friday, March 12, 2000, from 0800 through 1600;
Friday, March 19, 2000, from 0800 through 1600;
Friday, March 26, 2000, from 0800 through 1600;
Friday, April  2, 2000, from 0800 through 1600;
Friday, April  9, 2000, from 0800 through 1600;
Friday, April 16, 2000, from 0800 through 1600;
Friday, April 23, 2000, from 0800 through 1600;
Friday, April 30, 2000, from 0800 through 1600.
```

Properties not present in an instance of PolicyTimePeriodCondition
are implicitly treated as having their value "always enabled". Thus,
in the example above, the day-of-the-month mask is not present, and
so the validity period for the instance implicitly includes a day-
of-the-month mask that selects all days of the month. If we apply
this "missing property" rule to its fullest, we see that there is a
second way to indicate that a policy rule is always enabled: have it
point to an instance of PolicyTimePeriodCondition whose only
properties are its naming properties.

The property LocalOrUtcTime indicates whether the times represented
in the other five time-related properties of an instance of
PolicyTimePeriodCondition are to be interpreted as local times for
the location where a policy rule is being applied, or as UTC times.

The class definition is as follows.

```
NAME            PolicyTimePeriodCondition
DESCRIPTION     A class that provides the capability of enabling /
                disabling a policy rule according to a
                pre-determined schedule.
DERIVED FROM    PolicyCondition
ABSTRACT        FALSE
PROPERTIES      TimePeriod
                MonthOfYearMask
                DayOfMonthMask
                DayOfWeekMask
                TimeOfDayMask
                LocalOrUtcTime
```

6.5.1. The Property "TimePeriod"

   This property identifies an overall range of calendar dates and times
   over which a policy rule is valid. It reuses the format for an
   explicit time period defined in RFC 2445 (reference [10]): a string
   representing a starting date and time, in which the character 'T'
   indicates the beginning of the time portion, followed by the solidus
   character '/', followed by a similar string representing an end date
   and time. The first date indicates the beginning of the range, while
   the second date indicates the end. Thus, the second date and time
   must be later than the first. Date/times are expressed as substrings
   of the form "yyyymmddThhmmss". For example:

      20000101T080000/20000131T120000

         January 1, 2000, 0800 through January 31, 2000, noon

   There are also two special cases in which one of the date/time
   strings is replaced with a special string defined in RFC 2445.

   o If the first date/time is replaced with the string "THISANDPRIOR",
      then the property indicates that a policy rule is valid [from now]
      until the date/time that appears after the '/'.

   o If the second date/time is replaced with the string
      "THISANDFUTURE", then the property indicates that a policy rule
      becomes valid on the date/time that appears before the '/', and
      remains valid from that point on.

   Note that RFC 2445 does not use these two strings in connection with
   explicit time periods. Thus the PCIM is combining two elements from
   RFC 2445 that are not combined in the RFC itself.

   The property definition is as follows:

      NAME   TimePeriod
      DESCRIPTION   The range of calendar dates on which a policy
                      rule is valid.
      SYNTAX   string
      FORMAT   yyyymmddThhmmss/yyyymmddThhmmss, where the first
                      date/time may be replaced with the string
                      "THISANDPRIOR" or the second date/time may be
                      replaced with the string "THISANDFUTURE"

6.5.2. The Property "MonthOfYearMask"

   The purpose of this property is to refine the definition of the valid
   time period that is defined by the TimePeriod property, by explicitly
   specifying the months when the policy is valid. These properties
   work together, with the TimePeriod used to specify the overall time
   period during which the policy might be valid, and the
   MonthOfYearMask used to pick out the specific months within that time
   period when the policy is valid.

   This property is formatted as an octet string of size 2, consisting
   of 12 bits identifying the 12 months of the year, beginning with
   January and ending with December, followed by 4 bits that are always
   set to '0'. For each month, the value '1' indicates that the policy
   is valid for that month, and the value '0' indicates that it is not
   valid. The value X'08 30', for example, indicates that a policy rule
   is valid only in the months May, November, and December.

   See section 5.4 for details of how CIM represents a single-valued
   octet string property such as this one. (Basically, CIM prepends a
   4-octet length to the octet string.)

   If this property is omitted, then the policy rule is treated as valid
   for all twelve months. The property definition is as follows:

       NAME   MonthOfYearMask
       DESCRIPTION   A mask identifying the months of the year in
                       which a policy rule is valid.
       SYNTAX   octet string
       FORMAT   X'hh h0'

6.5.3. The Property "DayOfMonthMask"

   The purpose of this property is to refine the definition of the valid
   time period that is defined by the TimePeriod property, by explicitly
   specifying the days of the month when the policy is valid. These
   properties work together, with the TimePeriod used to specify the
   overall time period during which the policy might be valid, and the
   DayOfMonthMask used to pick out the specific days of the month within
   that time period when the policy is valid.

   This property is formatted as an octet string of size 8, consisting
   of 31 bits identifying the days of the month counting from the
   beginning, followed by 31 more bits identifying the days of the month
   counting from the end, followed by 2 bits that are always set to '0'.
   For each day, the value '1' indicates that the policy is valid for
   that day, and the value '0' indicates that it is not valid.

The value X'80 00 00 01 00 00 00 00', for example, indicates that a policy rule is valid on the first and last days of the month.

For months with fewer than 31 days, the digits corresponding to days that the months do not have (counting in both directions) are ignored.

The encoding of the 62 significant bits in the octet string matches that used for the schedDay object in the DISMAN-SCHEDULE-MIB. See reference [8] for more details on this object.

See section 5.4 for details of how CIM represents a single-valued octet string property such as this one. (Basically, CIM prepends a 4-octet length to the octet string.)

The property definition is as follows:

```
    NAME              DayOfMonthMask
    DESCRIPTION       A mask identifying the days of the month on
                      which a policy rule is valid.
    SYNTAX            octet string
    FORMAT            X'hh hh hh hh hh hh hh hh'
```

6.5.4. The Property "DayOfWeekMask"

The purpose of this property is to refine the definition of the valid time period that is defined by the TimePeriod property by explicitly specifying the days of the week when the policy is valid. These properties work together, with the TimePeriod used to specify the overall time period when the policy might be valid, and the DayOfWeekMask used to pick out the specific days of the week in that time period when the policy is valid.

This property is formatted as an octet string of size 1, consisting of 7 bits identifying the 7 days of the week, beginning with Sunday and ending with Saturday, followed by 1 bit that is always set to '0'. For each day of the week, the value '1' indicates that the policy is valid for that day, and the value '0' indicates that it is not valid.

The value X'7C', for example, indicates that a policy rule is valid Monday through Friday.

See section 5.4 for details of how CIM represents a single-valued octet string property such as this one. (Basically, CIM prepends a 4-octet length to the octet string.)

The property definition is as follows:

```
    NAME                DayOfWeekMask
    DESCRIPTION         A mask identifying the days of the week on which
                        a policy rule is valid.
    SYNTAX              octet string
    FORMAT              B'bbbb bbb0'
```

6.5.5. The Property "TimeOfDayMask"

The purpose of this property is to refine the definition of the valid
time period that is defined by the TimePeriod property by explicitly
specifying a range of times in a day the policy is valid for. These
properties work together, with the TimePeriod used to specify the
overall time period that the policy is valid for, and the
TimeOfDayMask used to pick out which range of time periods in a given
day of that time period the policy is valid for.

This property is formatted in the style of RFC 2445 [10]: a time
string beginning with the character 'T', followed by the solidus
character '/', followed by a second time string. The first time
indicates the beginning of the range, while the second time indicates
the end. Times are expressed as substrings of the form "Thhmmss".

The second substring always identifies a later time than the first
substring. To allow for ranges that span midnight, however, the
value of the second string may be smaller than the value of the first
substring. Thus, "T080000/T210000" identifies the range from 0800
until 2100, while "T210000/T080000" identifies the range from 2100
until 0800 of the following day.

When a range spans midnight, it by definition includes parts of two
successive days. When one of these days is also selected by either
the MonthOfYearMask, DayOfMonthMask, and/or DayOfWeekMask, but the
other day is not, then the policy is active only during the portion
of the range that falls on the selected day. For example, if the
range extends from 2100 until 0800, and the day of week mask selects
Monday and Tuesday, then the policy is active during the following
three intervals:

    From midnight Sunday until 0800 Monday;
    From 2100 Monday until 0800 Tuesday;
    From 2100 Tuesday until 23:59:59 Tuesday.

   The property definition is as follows:

      NAME   TimeOfDayMask
      DESCRIPTION   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.
      SYNTAX   string
      FORMAT   Thhmmss/Thhmmss

6.5.6. The Property "LocalOrUtcTime"

   This property indicates whether the times represented in the
   TimePeriod property and in the various Mask properties represent
   local times or UTC times. There is no provision for mixing of local
   times and UTC times: the value of this property applies to all of
   the other time-related properties.

   The property definition is as follows:

      NAME   LocalOrUtcTime
      DESCRIPTION   An indication of whether the other times in this
                         instance represent local times or UTC times.
      SYNTAX   uint16
      VALUES   localTime(1), utcTime(2)
      DEFAULT VALUE   utcTime(2)

6.6. The Class "VendorPolicyCondition"

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

   As its name suggests, this class is intended for vendor-specific
   extensions to the Policy Core Information Model. Standardized
   extensions are not expected to use this class.

   The class definition is as follows:

      NAME   VendorPolicyCondition
      DESCRIPTION   A class that defines a registered means to
                         describe a policy condition.
      DERIVED FROM   PolicyCondition
      ABSTRACT   FALSE
      PROPERTIES   Constraint[ ]
                         ConstraintEncoding

6.6.1. The Multi-valued Property "Constraint"

   This property provides a general extension mechanism for representing
   policy conditions that have not been modeled with specific
   properties. The format of the octet strings in the array is left
   unspecified in this definition. It is determined by the OID value
   stored in the property ConstraintEncoding. Since ConstraintEncoding
   is single-valued, all the values of Constraint share the same format
   and semantics.

   See Section 5.4 for a description of how CIM encodes an array of
   octet strings like this one.

   A policy decision point can readily determine whether it supports the
   values stored in an instance of Constraint by checking the OID value
   from ConstraintEncoding against the set of OIDs it recognizes. The
   action for the policy decision point to take in case it does not
   recognize the format of this data could itself be modeled as a policy
   rule, governing the behavior of the policy decision point.

   The property is defined as follows:

       NAME   Constraint
       DESCRIPTION   Extension mechanism for representing constraints
                       that have not been modeled as specific
                       properties. The format of the values is
                       identified by the OID stored in the property
                       ConstraintEncoding.
       SYNTAX   octet string

6.6.2. The Property "ConstraintEncoding"

   This property identifies the encoding and semantics of the Constraint
   property values in this instance. The value of this property is a
   single string, representing a single OID.

   The property is defined as follows:

       NAME   ConstraintEncoding
       DESCRIPTION   An OID encoded as a string, identifying the format
                       and semantics for this instance's Constraint
                       property. The value is a dotted sequence of
                       decimal digits (for example, "1.2.100.200")
                       representing the arcs of the OID. The characters
                       in the string are the UCS-2 characters
                       corresponding to the US ASCII encodings of the
                       numeric characters and the period.
       SYNTAX   string

6.7. The Abstract Class "PolicyAction"

   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 state. This (new) state provides one or
   more (new) behaviors. A policy action ordinarily changes the
   configuration of one or more elements.

   A PolicyRule contains one or more policy actions. A policy
   administrator can assign an order to the actions associated with a
   PolicyRule, complete with an indication of whether the indicated
   order is mandatory, recommended, or of no significance. Ordering of
   the actions associated with a PolicyRule is accomplished via a
   property in the PolicyActionInPolicyRule aggregation.

   The actions associated with a PolicyRule are executed if and only if
   the overall condition(s) of the PolicyRule evaluates to TRUE.

   The class definition of PolicyAction is as follows:

        NAME   PolicyAction
        DESCRIPTION   A class representing a rule-specific or reusable
                        policy action to be performed if the condition for
                        a policy rule evaluates to TRUE.
        DERIVED FROM   Policy
        ABSTRACT   TRUE
        PROPERTIES   NONE

   No properties are defined for this class since it inherits all its
   properties from Policy. The class exists as an abstract superclass
   for domain-specific policy actions, defined in subclasses. In an
   implementation, various key/identification properties MUST be defined
   for the class or its instantiable subclasses. The keys for a native
   CIM implementation are defined in Appendix A, Section 13.3. Keys for
   an LDAP implementation will be defined in the LDAP mapping of this
   information model [11].

   When identifying and using the PolicyAction class, it is necessary to
   remember that an action can be rule-specific or reusable. This was
   discussed above in Section 5.1. The distinction between the two
   types of policy actions lies in the associations in which an instance
   can participate, and in how the different instances are named.
   Conceptually, a reusable policy action resides in a policy
   repository, and is named within the scope of that repository. On the
   other hand, a rule-specific policy action is named within the scope
   of the single policy rule to which it is related.

The distinction between rule-specific and reusable PolicyActions
affects the CIM naming, defined in Appendix A, and the LDAP mapping
[11].

6.8. The Class "VendorPolicyAction"

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, the two properties ActionData and
ActionEncoding 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 to the Policy Core Information Model. Standardized
extensions are not expected to use this class.

The class definition is as follows:

```
NAME               VendorPolicyAction
DESCRIPTION        A class that defines a registered means to
                   describe a policy action.
DERIVED FROM       PolicyAction
ABSTRACT           FALSE
PROPERTIES         ActionData[ ]
                   ActionEncoding
```

6.8.1. The Multi-valued Property "ActionData"

This property provides a general extension mechanism for representing
policy actions that have not been modeled with specific properties.
The format of the octet strings in the array is left unspecified in
this definition. It is determined by the OID value stored in the
property ActionEncoding. Since ActionEncoding is single-valued, all
the values of ActionData share the same format and semantics. See
Section 5.4 for a discussion of how CIM encodes an array of octet
strings like this one.

A policy decision point can readily determine whether it supports the
values stored in an instance of ActionData by checking the OID value
from ActionEncoding against the set of OIDs it recognizes. The
action for the policy decision point to take in case it does not
recognize the format of this data could itself be modeled as a policy
rule, governing the behavior of the policy decision point.

The property is defined as follows:

    NAME   ActionData
    DESCRIPTION   Extension mechanism for representing actions that
                    have not been modeled as specific properties. The
                    format of the values is identified by the OID
                    stored in the property ActionEncoding.
    SYNTAX   octet string

6.8.2. The Property "ActionEncoding"

    This property identifies the encoding and semantics of the ActionData
    property values in this instance. The value of this property is a
    single string, representing a single OID.

    The property is defined as follows:

    NAME   ActionEncoding
    DESCRIPTION   An OID encoded as a string, identifying the format
                    and semantics for this instance's ActionData
                    property. The value is a dotted sequence of
                    decimal digits (for example, "1.2.100.200")
                    representing the arcs of the OID. The characters
                    in the string are the UCS-2 characters
                    corresponding to the US ASCII encodings of the
                    numeric characters and the period.
    SYNTAX   string

6.9. The Class "PolicyRepository"

    The class definition of PolicyRepository is as follows:

    NAME   PolicyRepository
    DESCRIPTION   A class representing an administratively defined
                    container for reusable policy-related
                    information. This class does not introduce any
                    additional properties beyond those in its
                    superclass AdminDomain. It does, however,
                    participate in a number of unique associations.
    DERIVED FROM   AdminDomain
    ABSTRACT   FALSE

7. Association and Aggregation Definitions

    The first two subsections of this section introduce associations and
    aggregations as they are used in CIM. The remaining subsections
    present the class definitions for the associations and aggregations
    that are part of the Policy Core Information Model.

7.1. Associations

   An association is a CIM construct representing a relationship between
   two (or theoretically more) objects. It is modeled as a class
   containing typically two object references. Associations can be
   defined between classes without affecting any of the related classes.
   That is, addition of an association does not affect the interface of
   the related classes.

7.2. Aggregations

   An aggregation is a strong form of an association, which usually
   represents a "whole-part" or a "collection" relationship. For
   example, CIM uses an aggregation to represent the containment
   relationship between a system and the components that make up the
   system. Aggregation as a "whole-part" relationship often implies,
   but does not require, that the aggregated objects have mutual
   dependencies.

7.3. The Abstract Aggregation "PolicyComponent

   This abstract aggregation defines two object references that will be
   overridden in each of five subclasses, to become references to the
   concrete policy classes PolicyGroup, PolicyRule, PolicyCondition,
   PolicyAction, and PolicyTimePeriodCondition. The value of the
   abstract superclass is to convey that all five subclasses have the
   same "whole- part" semantics, and for ease of query to locate all
   "components" of a PolicyGroup or PolicyRule.

   The class definition for the aggregation is as follows:

      NAME   PolicyComponent
      DESCRIPTION   A generic aggregation used to establish 'part of'
                      relationships between the subclasses of
                      Policy. For example, the
                      PolicyConditionInPolicyRule aggregation defines
                      that PolicyConditions are part of a PolicyRule.
      ABSTRACT   TRUE
      PROPERTIES   GroupComponent[ref Policy[0..n]]
                      PartComponent[ref Policy[0..n]]

7.4. The Aggregation "PolicyGroupInPolicyGroup"

   The PolicyGroupInPolicyGroup aggregation enables policy groups to be
   nested. This is critical for scalability and manageability, as it
   enables complex policies to be constructed from multiple simpler

policies for administrative convenience. For example, a policy group
representing policies for the US might have nested within it policy
groups for the Eastern and Western US.

A PolicyGroup may aggregate other PolicyGroups via this aggregation,
or it may aggregate PolicyRules via the PolicyRuleInPolicyGroup
aggregation. Note that it is assumed that this aggregation is used
to form directed acyclic graphs and NOT ring structures.The class
definition for the aggregation is as follows:

```
    NAME              PolicyGroupInPolicyGroup
    DESCRIPTION       A class representing the aggregation of
                      PolicyGroups by a higher-level PolicyGroup.
    DERIVED FROM      PolicyComponent
    ABSTRACT          FALSE
    PROPERTIES        GroupComponent[ref PolicyGroup[0..n]]
                      PartComponent[ref PolicyGroup[0..n]]
```

7.4.1. The Reference "GroupComponent"

This property is inherited from PolicyComponent, and overridden to
become an object reference to a PolicyGroup that contains one or more
other PolicyGroups. Note that for any single instance of the
aggregation class PolicyGroupInPolicyGroup, this property (like all
Reference properties) is single-valued. The [0..n] cardinality
indicates that there may be 0, 1, or more than one PolicyGroups that
contain any given PolicyGroup.

7.4.2. The Reference "PartComponent"

This property is inherited from PolicyComponent, and overridden to
become an object reference to a PolicyGroup contained by one or more
other PolicyGroups. Note that for any single instance of the
aggregation class PolicyGroupInPolicyGroup, this property (like all
Reference properties) is single-valued. The [0..n] cardinality
indicates that a given PolicyGroup may contain 0, 1, or more than one
other PolicyGroups.

7.5. The Aggregation "PolicyRuleInPolicyGroup"

A policy group may aggregate one or more policy rules, via the
PolicyRuleInPolicyGroup aggregation. Grouping of policy rules into a
policy group is again for administrative convenience; a policy rule
may also be used by itself, without belonging to a policy group.

A PolicyGroup may aggregate PolicyRules via this aggregation, or it
may aggregate other PolicyGroups via the PolicyGroupInPolicyGroup
aggregation.

The class definition for the aggregation is as follows:

```
    NAME              PolicyRuleInPolicyGroup
    DESCRIPTION       A class representing the aggregation of
                      PolicyRules by a PolicyGroup.
    DERIVED FROM      PolicyComponent
    ABSTRACT          FALSE
    PROPERTIES        GroupComponent[ref PolicyGroup[0..n]]
                      PartComponent[ref PolicyRule[0..n]]
```

7.5.1. The Reference "GroupComponent"

   This property is inherited from PolicyComponent, and overridden to
   become an object reference to a PolicyGroup that contains one or more
   PolicyRules. Note that for any single instance of the aggregation
   class PolicyRuleInPolicyGroup, this property (like all Reference
   properties) is single-valued. The [0..n] cardinality indicates that
   there may be 0, 1, or more than one PolicyGroups that contain any
   given PolicyRule.

7.5.2. The Reference "PartComponent"

   This property is inherited from PolicyComponent, and overridden to
   become an object reference to a PolicyRule contained by one or more
   PolicyGroups. Note that for any single instance of the aggregation
   class PolicyRuleInPolicyGroup, this property (like all Reference
   properties) is single-valued. The [0..n] cardinality indicates that
   a given PolicyGroup may contain 0, 1, or more than one PolicyRules.

7.6. The Aggregation "PolicyConditionInPolicyRule"

   A policy rule aggregates zero or more instances of the
   PolicyCondition class, via the PolicyConditionInPolicyRule
   association. A policy rule that aggregates zero policy conditions
   must indicate in its class definition what "triggers" the performance
   of its actions. In short, it must describe its implicit
   PolicyConditions, since none are explicitly associated. For example,
   there might be a subclass of PolicyRule named "HttpPolicyRule", where
   the class definition assumes that the condition, "If HTTP traffic,"
   is true before the rule's actions would be performed. There is no
   need to formalize and instantiate this condition, since it is obvious
   in the semantics of the PolicyRule.

   The conditions aggregated by a policy rule are grouped into two
   levels of lists: either an ORed set of ANDed sets of conditions (DNF,
   the default) or an ANDed set of ORed sets of conditions (CNF).
   Individual conditions in these lists may be negated. The property
   ConditionListType (in PolicyRule) specifies which of these two

   grouping schemes applies to a particular PolicyRule. The conditions
   are used to determine whether to perform the actions associated with
   the PolicyRule.

   One or more policy time periods may be among the conditions
   associated with a policy rule via the PolicyConditionInPolicyRule
   association. In this case, the time periods are simply additional
   conditions to be evaluated along with any other conditions specified
   for the rule.

   The class definition for the aggregation is as follows:

```
    NAME            PolicyConditionInPolicyRule
    DESCRIPTION     A class representing the aggregation of
                    PolicyConditions by a PolicyRule.
    DERIVED FROM    PolicyComponent
    ABSTRACT        FALSE
    PROPERTIES      GroupComponent[ref PolicyRule[0..n]]
                    PartComponent[ref PolicyCondition[0..n]]
                    GroupNumber
                    ConditionNegated
```

7.6.1. The Reference "GroupComponent"

   This property is inherited from PolicyComponent, and overridden to
   become an object reference to a PolicyRule that contains one or more
   PolicyConditions. Note that for any single instance of the
   aggregation class PolicyConditionInPolicyRule, this property (like
   all Reference properties) is single-valued. The [0..n] cardinality
   indicates that there may be 0, 1, or more than one PolicyRules that
   contain any given PolicyCondition.

7.6.2. The Reference "PartComponent"

   This property is inherited from PolicyComponent, and overridden to
   become an object reference to a PolicyCondition contained by one or
   more PolicyRules. Note that for any single instance of the
   aggregation class PolicyConditionInPolicyRule, this property (like
   all Reference properties) is single-valued. The [0..n] cardinality
   indicates that a given PolicyRule may contain 0, 1, or more than one
   PolicyConditions.

7.6.3. The Property "GroupNumber"

   This property contains an integer identifying the group to which the
   condition referenced by the PartComponent property is assigned in
   forming the overall conditional expression for the policy rule
   identified by the GroupComponent reference.

   The property is defined as follows:

        NAME   GroupNumber
        DESCRIPTION   Unsigned integer indicating the group to which
                            the condition identified by the PartComponent
                            property is to be assigned.
        SYNTAX   uint16
        DEFAULT   0

7.6.4. The Property "ConditionNegated"

   This property is a boolean, indicating whether the condition
   referenced by the PartComponent property is negated in forming the
   overall conditional expression for the policy rule identified by the
   GroupComponent reference.

   The property is defined as follows:

        NAME   ConditionNegated
        DESCRIPTION   Indication of whether the condition identified by
                            the PartComponent property is negated. (TRUE
                            indicates that the condition is negated, FALSE
                            indicates that it is not negated.)
        SYNTAX   boolean
        DEFAULT   FALSE

7.7. The Aggregation "PolicyRuleValidityPeriod"

   A different relationship between a policy rule and a policy time
   period (than PolicyConditionInPolicyRule) is represented by the
   PolicyRuleValidityPeriod aggregation. The latter describes scheduled
   activation and deactivation of the policy rule.

   If a policy rule is associated with multiple policy time periods via
   this association, then the rule is active if at least one of the time
   periods indicates that it is active. (In other words, the time
   periods are ORed to determine whether the rule is active.) A policy
   time period may be aggregated by multiple policy rules. A rule that
   does not point to a policy time period via this aggregation is, from
   the point of view of scheduling, always active. It may, however, be
   inactive for other reasons.

   Time periods are a general concept that can be used in other
   applications. However, they are mentioned explicitly here in this
   specification since they are frequently used in policy applications.

The class definition for the aggregation is as follows:

```
NAME               PolicyRuleValidityPeriod
DESCRIPTION        A class representing the aggregation of
                   PolicyTimePeriodConditions by a PolicyRule.
DERIVED FROM       PolicyComponent
ABSTRACT           FALSE
PROPERTIES         GroupComponent[ref PolicyRule[0..n]]
                   PartComponent[ref PolicyTimePeriodCondition[0..n]]
```

7.7.1. The Reference "GroupComponent"

   This property is inherited from PolicyComponent, and overridden to
   become an object reference to a PolicyRule that contains one or more
   PolicyTimePeriodConditions. Note that for any single instance of the
   aggregation class PolicyRuleValidityPeriod, this property (like all
   Reference properties) is single-valued. The [0..n] cardinality
   indicates that there may be 0, 1, or more than one PolicyRules that
   contain any given PolicyTimePeriodCondition.

7.7.2. The Reference "PartComponent"

   This property is inherited from PolicyComponent, and overridden to
   become an object reference to a PolicyTimePeriodCondition contained
   by one or more PolicyRules. Note that for any single instance of the
   aggregation class PolicyRuleValidityPeriod, this property (like all
   Reference properties) is single-valued. The [0..n] cardinality
   indicates that a given PolicyRule may contain 0, 1, or more than one
   PolicyTimePeriodConditions.

7.8. The Aggregation "PolicyActionInPolicyRule"

   A policy rule may aggregate zero or more policy actions. A policy
   rule that aggregates zero policy actions must indicate in its class
   definition what actions are taken when the rule's conditions evaluate
   to TRUE. In short, it must describe its implicit PolicyActions,
   since none are explicitly associated. For example, there might be a
   subclass of PolicyRule representing a Diffserv absolute dropper,
   where the subclass itself indicates the action to be taken. There is
   no need to formalize and instantiate this action, since it is obvious
   in the semantics of the PolicyRule.

   The actions associated with a PolicyRule may be given a required
   order, a recommended order, or no order at all. For actions
   represented as separate objects, the PolicyActionInPolicyRule
   aggregation can be used to express an order.

This aggregation does not indicate whether a specified action order
is required, recommended, or of no significance; the property
SequencedActions in the aggregating instance of PolicyRule provides
this indication.

The class definition for the aggregation is as follows:

```
    NAME              PolicyActionInPolicyRule
    DESCRIPTION       A class representing the aggregation of
                      PolicyActions by a PolicyCondition.
    DERIVED FROM      PolicyComponent
    ABSTRACT          FALSE
    PROPERTIES        GroupComponent[ref PolicyRule[0..n]]
                      PartComponent[ref PolicyAction[0..n]]
                      ActionOrder
```

7.8.1. The Reference "GroupComponent"

This property is inherited from PolicyComponent, and overridden to
become an object reference to a PolicyRule that contains one or more
PolicyActions. Note that for any single instance of the aggregation
class PolicyActionInPolicyRule, this property (like all Reference
properties) is single-valued. The [0..n] cardinality indicates that
there may be 0, 1, or more than one PolicyRules that contain any
given PolicyAction.

7.8.2. The Reference "PartComponent"

This property is inherited from PolicyComponent, and overridden to
become an object reference to a PolicyAction contained by one or more
PolicyRules. Note that for any single instance of the aggregation
class PolicyActionInPolicyRule, this property (like all Reference
properties) is single-valued. The [0..n] cardinality indicates that
a given PolicyRule may contain 0, 1, or more than one PolicyActions.

7.8.3. The Property "ActionOrder"

This property provides an unsigned integer 'n' that indicates the
relative position of an action in the sequence of actions associated
with a policy rule. When 'n' is a positive integer, it indicates a
place in the sequence of actions to be performed, with smaller
integers indicating earlier positions in the sequence. The special
value '0' indicates "don't care". If two or more actions have the
same non-zero sequence number, they may be performed in any order,
but they must all be performed at the appropriate place in the
overall action sequence.

A series of examples will make ordering of actions clearer:

o If all actions have the same sequence number, regardless of
   whether it is '0' or non-zero, any order is acceptable.

o The values

   1:ACTION A
   2:ACTION B
   1:ACTION C
   3:ACTION D

   indicate two acceptable orders: A,C,B,D or C,A,B,D, since A and C
   can be performed in either order, but only at the '1' position.

o The values

   0:ACTION A
   2:ACTION B
   3:ACTION C
   3:ACTION D

   require that B,C, and D occur either as B,C,D or as B,D,C. Action
   A may appear at any point relative to B,C, and D. Thus the
   complete set of acceptable orders is: A,B,C,D; B,A,C,D; B,C,A,D;
   B,C,D,A; A,B,D,C; B,A,D,C; B,D,A,C; B,D,C,A.

   Note that the non-zero sequence numbers need not start with '1',
   and they need not be consecutive. All that matters is their
   relative magnitude.

   The property is defined as follows:

   NAME   ActionOrder
   DESCRIPTION   Unsigned integer indicating the relative position
                   of an action in the sequence of actions aggregated
                   by a policy rule.
   SYNTAX   uint16

7.9. The Abstract Association "PolicyInSystem"

   This abstract association inherits two object references from a
   higher- level CIM association class, Dependency. It overrides these
   object references to make them references to instances of the classes
   System and Policy. Subclasses of PolicyInSystem then override these
   object references again, to make them references to concrete policy
   classes.

The value of the abstract superclass is to convey that all subclasses
have the same "dependency" semantics, and for ease of query to locate
all policy "dependencies" on a System. These dependencies are
related to scoping or hosting of the Policy.

The class definition for the association is as follows:

```
NAME            PolicyInSystem
DESCRIPTION     A generic association used to establish
                dependency relationships between Policies and the
                Systems that host them.
DERIVED FROM    Dependency
ABSTRACT        TRUE
PROPERTIES      Antecedent[ref System[0..1]]
                Dependent[ref Policy[0..n]]
```

7.10. The Weak Association "PolicyGroupInSystem"

   This association links a PolicyGroup to the System in whose scope the
   PolicyGroup is defined.

   The class definition for the association is as follows:

```
NAME            PolicyGroupInSystem
DESCRIPTION     A class representing the fact that a PolicyGroup
                is defined within the scope of a System.
DERIVED FROM    PolicyInSystem
ABSTRACT        FALSE
PROPERTIES      Antecedent[ref System[1..1]]
                Dependent[ref PolicyGroup[weak]]
```

7.10.1. The Reference "Antecedent"

   This property is inherited from PolicyInSystem, and overridden to
   restrict its cardinality to [1..1]. It serves as an object reference
   to a System that provides a scope for one or more PolicyGroups.
   Since this is a weak association, the cardinality for this object
   reference is always 1, that is, a PolicyGroup is always defined
   within the scope of exactly one System.

7.10.2. The Reference "Dependent"

   This property is inherited from PolicyInSystem, and overridden to
   become an object reference to a PolicyGroup defined within the scope
   of a System. Note that for any single instance of the association
   class PolicyGroupInSystem, this property (like all Reference

properties) is single-valued. The [0..n] cardinality indicates that
a given System may have 0, 1, or more than one PolicyGroups defined
within its scope.

7.11. The Weak Association "PolicyRuleInSystem"

Regardless of whether it belongs to a PolicyGroup (or to multiple
PolicyGroups), a PolicyRule is itself defined within the scope of a
System. This association links a PolicyRule to the System in whose
scope the PolicyRule is defined.

The class definition for the association is as follows:

```
NAME               PolicyRuleInSystem
DESCRIPTION        A class representing the fact that a PolicyRule
                   is defined within the scope of a System.
DERIVED FROM       PolicyInSystem
ABSTRACT           FALSE
PROPERTIES         Antecedent[ref System[1..1]]
                   Dependent[ref PolicyRule[weak]]
```

7.11.1. The Reference "Antecedent"

This property is inherited from PolicyInSystem, and overridden to
restrict its cardinality to [1..1]. It serves as an object reference
to a System that provides a scope for one or more PolicyRules. Since
this is a weak association, the cardinality for this object reference
is always 1, that is, a PolicyRule is always defined within the scope
of exactly one System.

7.11.2. The Reference "Dependent"

This property is inherited from PolicyInSystem, and overridden to
become an object reference to a PolicyRule defined within the scope
of a System. Note that for any single instance of the association
class PolicyRuleInSystem, this property (like all Reference
properties) is single-valued. The [0..n] cardinality indicates that
a given System may have 0, 1, or more than one PolicyRules defined
within its scope.

7.12. The Association "PolicyConditionInPolicyRepository"

A reusable policy condition is always related to a single
PolicyRepository, via the PolicyConditionInPolicyRepository
association. This is not true for all PolicyConditions, however. An
instance of PolicyCondition that represents a rule-specific condition
is not related to any policy repository via this association.

The class definition for the association is as follows:

```
    NAME              PolicyConditionInPolicyRepository
    DESCRIPTION       A class representing the inclusion of a reusable
                      PolicyCondition in a PolicyRepository.
    DERIVED FROM      PolicyInSystem
    ABSTRACT          FALSE
    PROPERTIES        Antecedent[ref PolicyRepository[0..1]]
                      Dependent[ref PolicyCondition[0..n]]
```

7.12.1. The Reference "Antecedent"

   This property is inherited from PolicyInSystem, and overridden to
   become an object reference to a PolicyRepository containing one or
   more PolicyConditions. A reusable PolicyCondition is always related
   to exactly one PolicyRepository via the
   PolicyConditionInPolicyRepository association. The [0..1]
   cardinality for this property covers the two types of
   PolicyConditions: 0 for a rule-specific PolicyCondition, 1 for a
   reusable one.

7.12.2. The Reference "Dependent"

   This property is inherited from PolicyInSystem, and overridden to
   become an object reference to a PolicyCondition included in a
   PolicyRepository. Note that for any single instance of the
   association class PolicyConditionInPolicyRepository, this property
   (like all Reference properties) is single-valued. The [0..n]
   cardinality indicates that a given PolicyRepository may contain 0, 1,
   or more than one PolicyConditions.

7.13. The Association "PolicyActionInPolicyRepository"

   A reusable policy action is always related to a single
   PolicyRepository, via the PolicyActionInPolicyRepository association.
   This is not true for all PolicyActions, however. An instance of
   PolicyAction that represents a rule-specific action is not related to
   any policy repository via this association.

   The class definition for the association is as follows:

```
    NAME              PolicyActionInPolicyRepository
    DESCRIPTION       A class representing the inclusion of a reusable
                      PolicyAction in a PolicyRepository.
    DERIVED FROM      PolicyInSystem
    ABSTRACT          FALSE
    PROPERTIES        Antecedent[ref PolicyRepository[0..1]]
                      Dependent[ref PolicyAction[0..n]]
```

7.13.1. The Reference "Antecedent"

   This property is inherited from PolicyInSystem, and overridden to
   become an object reference to a PolicyRepository containing one or
   more PolicyActions. A reusable PolicyAction is always related to
   exactly one PolicyRepository via the PolicyActionInPolicyRepository
   association. The [0..1] cardinality for this property covers the two
   types of PolicyActions: 0 for a rule-specific PolicyAction, 1 for a
   reusable one.

7.13.2. The Reference "Dependent"

   This property is inherited from PolicyInSystem, and overridden to
   become an object reference to a PolicyAction included in a
   PolicyRepository. Note that for any single instance of the
   association class PolicyActionInPolicyRepository, this property (like
   all Reference properties) is single-valued. The [0..n] cardinality
   indicates that a given PolicyRepository may contain 0, 1, or more
   than one PolicyActions.

7.14. The Aggregation "PolicyRepositoryInPolicyRepository"

   The PolicyRepositoryInPolicyRepository aggregation enables policy
   repositories to be nested. This derives from the higher level CIM
   association, CIM_SystemComponent, describing that Systems contain
   other ManagedSystemElements. This superclass could not be used for
   the other Policy aggregations, since Policies are not
   ManagedSystemElements, but ManagedElements. Note that it is assumed
   that this aggregation is used to form directed acyclic graphs and NOT
   ring structures.

   The class definition for the aggregation is as follows:

        NAME   PolicyRepositoryInPolicyRepository
        DESCRIPTION   A class representing the aggregation of
                          PolicyRepositories by a higher-level
                          PolicyRepository.
        DERIVED FROM   SystemComponent
        ABSTRACT   FALSE
        PROPERTIES   GroupComponent[ref PolicyRepository[0..n]]
                            PartComponent[ref PolicyRepository[0..n]]

7.14.1. The Reference "GroupComponent"

   This property is inherited from the CIM class SystemComponent, and
   overridden to become an object reference to a PolicyRepository that
   contains one or more other PolicyRepositories. Note that for any
   single instance of the aggregation class
   PolicyRepositoryInPolicyRepository, this property (like all Reference

properties) is single-valued. The [0..n] cardinality indicates that
there may be 0, 1, or more than one PolicyRepositories that contain
any given PolicyRepository.

7.14.2. The Reference "PartComponent"

This property is inherited from the CIM class SystemComponent, and
overridden to become an object reference to a PolicyRepository
contained by one or more other PolicyRepositories. Note that for any
single instance of the aggregation class
PolicyRepositoryInPolicyRepository, this property (like all Reference
properties) is single-valued. The [0..n] cardinality indicates that
a given PolicyRepository may contain 0, 1, or more than one other
PolicyRepositories.

8. Intellectual Property

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

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

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

9. Acknowledgements

The Policy Core Information Model in this document is closely based
on the work of the DMTF's Service Level Agreements working group, so
thanks are due to the members of that working group. Several of the
policy classes in this model first appeared in early 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. Some other elements

10. Security Considerations

   The Policy Core Information Model (PCIM) presented in this document
   provides an object-oriented model for describing policy information.
   It provides a basic framework for describing the structure of policy
   information, in a form independent of any specific repository or
   access protocol, for use by an operational system. PCIM is not
   intended to represent any particular system design or implementation,
   nor does it define a protocol, and as such it does not have any
   specific security requirements.

   However, it should also be noted that certain derivative documents,
   which use PCIM as a base, will need to convey more specific security
   considerations. In order to communicate the nature of what will be
   expected in these follow-on derivative documents, it is necessary to
   review the reasons that PCIM, as defined in this document, is neither
   implementable, nor representative of any real-world system, as well
   as the nature of the expected follow-on extensions and mappings.

   There are three independent reasons that PCIM, as defined here, is
   neither implementable nor representative of any real-world system:

   1. Its classes are independent of any specific repository that
      uses any specific access protocol. Therefore, its classes are
      designed not to be implemented directly. PCIM should instead
      be viewed as a schematic that directs how information should be
      represented, independent of any specific model implementation
      constraints.

   2. Its classes were designed to be independent of any specific
      policy domain. For example, DiffServ and IPSec represent two
      different policy domains. Each document which 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.

   3. It's an information model, which must be mapped to a specific
      data model (native CIM schema, LDAP schema, MIB, whatever)
      before it can be implemented. Derivative documents will map
      the extended information models noted in item 2, above, to
      specific types of data model implementations.

Even though specific security requirements are not appropriate for
PCIM, 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 PCIM may
have extensive security requirements that impact nearly all classes
and subclasses 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 which derive from PCIM.

For example, in some real-world scenarios, the values associated with
certain properties, within certain instantiated 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.
As long as the derived model remains an information model (as opposed
to a data model), it is not possible to discuss 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. Therefore,
these mechanisms will need to be discussed in each of the data models
to which the derived information models are mapped. If there are any
general security requirements that can be identified and can be
applied across multiple types of data models, it would be appropriate
to discuss those at the information model level, rather than the data
model level. In any case, any identified security requirements that
are not dealt with in the information model document, MUST be dealt
with in the derivative data model documents.

We can illustrate these points by extending the example from Section
2. A real-world system that provides QoS Gold Service to John would
likely need to provide at least the following security-related
capabilities and mechanisms (see [12] for definitions of security
related terms):

o Data integrity for the information (e.g., property values and
   instantiated relationships) that specify that John gets QoS Gold
   Service, from the point(s) that the information is entered into
   the system to the point(s) where network components actually
   provide that Service.

o Authentication and Authorization methods to ensure that only
   system administrators (and not John or other engineers) can
   remotely administer components of the system.

   o An Authentication method to insure that John receives Gold
      Service, and the other members of the engineering group receive
      Bronze Service.

   These are one possible set of requirements associated with an example
   real-world system which delivers Gold Service, and the appropriate
   place to document these would be in some combination of the
   information model and the derivative data models for QoS Policy.
   Each of the data models would also need to discuss how these
   requirements are satisfied, using the mechanisms typically available
   to such a data model, given the particular technology or set of
   technologies which it may employ.

11. References

   [1] Distributed Management Task Force, Inc., "DMTF Technologies: CIM
        Standards << CIM Schema: Version 2.4", available via links on
        the following DMTF web page:
        http://www.dmtf.org/spec/cim_schema_v24.html.

   [2] Distributed Management Task Force, Inc., "Common Information
        Model (CIM) Specification, version 2.2, June 1999. This
        document is available on the following DMTF web page:
        http://www.dmtf.org/spec/cims.html.

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

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

   [5] J. Strassner and S. Judd, "Directory-Enabled Networks", version
        3.0c5 (August 1998). A PDF file is available at
        http://www.murchiso.com/den/#denspec.

   [6] J. Strassner, 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.

   [7] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC
        2279, January 1998.

   [8] Levi, D. and J. Schoenwaelder, "Definitions of Managed Objects
        for Scheduling Management Operations", RFC 2591, May 1999.

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

   [10] Dawson, F. and D. Stenerson, "Internet Calendaring and
        Scheduling Core Object Specification (iCalendar)", RFC 2445,
        November 1998.

   [11] Strassner, J., and E. Ellesson, B. Moore, R. Moats, "Policy Core
        LDAP Schema", Work in Progress.

   [12] Shirey, R., "Internet Security Glossary", FYI 36, RFC 2828, May
        2000.

   Note: the CIM 2.4 Schema specification is defined by the following
   set of MOF files, available from the following URL:

      http://www.dmtf.org/spec/CIM_Schema24/CIM_Schema24.zip

12. Authors' Addresses

Ed Ellesson
LongBoard, Inc.
2505 Meridian Pkwy, #100
Durham, NC 27713

Phone: +1 919-361-3230
Fax: +1 919-361-3299
EMail: eellesson@lboard.com

Bob Moore
IBM Corporation, BRQA/502
4205 S. Miami Blvd.
Research Triangle Park, NC 27709

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

John Strassner
Cisco Systems, Bldg 15
170 West Tasman Drive
San Jose, CA 95134

Phone: +1 408-527-1069
Fax: +1 408-527-6351
EMail: johns@cisco.com

Andrea Westerinen
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134

Phone: +1 408-853-8294
Fax: +1 408-527-6351
EMail: andreaw@cisco.com

13. Appendix A: Class Identification in a Native CIM Implementation

   While the CommonName property is present in the abstract superclass
   Policy, and is thus available in all of its instantiable subclasses,
   CIM does not use this property for naming instances. The following
   subsections discuss how naming is handled in a native CIM
   implementation for each of the instantiable classes in the Policy
   Core Information Model.

   Two things should be noted regarding CIM naming:

   o When a CIM association is specified as "weak", this is a statement
      about naming scopes: an instance of the class at the weak end of
      the association is named within the scope of an instance of the
      class at the other end of the association. This is accomplished
      by propagation of keys from the instance of the scoping class to
      the instance of the weak class. Thus the weak class has, via key
      propagation, all the keys from the scoping class, and it also has
      one or more additional keys for distinguishing instances of the
      weak class, within the context of the scoping class.

   o All class names in CIM are limited to alphabetic and numeric
      characters plus the underscore, with the restriction that the
      first character cannot be numeric. Refer to Appendix F "Unicode
      Usage" in reference [2] for an exact specification of how CIM
      class names are encoded in CIM strings.

13.1. Naming Instances of PolicyGroup and PolicyRule

   A policy group always exists in the context of a system. In the
   Policy Core Information Model, this is captured by the weak
   aggregation PolicyGroupInSystem between a PolicyGroup and a System.
   Note that System serves as the base class for describing network
   devices and administrative domains.

   A policy rule also exists in the context of a system. In the Policy
   Core Information Model, this is captured by the weak association
   PolicyRuleInSystem between a PolicyRule and a System.

   The following sections define the CIM keys for PolicyGroup and
   PolicyRule.

13.1.1. PolicyGroup's CIM Keys

   The CIM keys of the PolicyGroup class are:

   o SystemCreationClassName (A CIM_System key, propagated due to the
      weak association, PolicyGroupInSystem)

       o SystemName (A CIM_System key, propagated due to the weak
          association, PolicyGroupInSystem)
       o CreationClassName
       o PolicyGroupName

       They are defined in Reference [1] as follows:

       NAME   SystemCreationClassName
       DESCRIPTION   SystemCreationClassName represents the class name of
                         the CIM System object providing the naming scope for
                         the instance of PolicyGroup.
       SYNTAX   string [MaxLen 256]
       QUALIFIER   key

       NAME   SystemName
       DESCRIPTION   SystemName represent the individual name of the
                         particular System object, providing the naming scope
                         for the instance of PolicyGroup.
       SYNTAX   string [MaxLen 256]
       QUALIFIER   key

       NAME   CreationClassName
       DESCRIPTION   This property is set to "CIM_PolicyGroup", if the
                         PolicyGroup object is directly instantiated. Or, it
                         is equal to the class name of the PolicyGroup
                         subclass that is instantiated.
       SYNTAX   string [MaxLen 256]
       QUALIFIER   key

       NAME   PolicyGroupName
       DESCRIPTION   The identifying name of this policy group.
       SYNTAX   string [MaxLen 256]
       QUALIFIER   key

13.1.2. PolicyRule's CIM Keys

       The CIM keys of the PolicyRule class are:

       o SystemCreationClassName (A CIM_System key, propagated due to the
          weak association PolicyRuleInSystem)
       o SystemName (A CIM_System key, propagated due to the weak
          association PolicyRuleInSystem)
       o CreationClassName
       o PolicyRuleName

       SystemCreationClassName and SystemName work the same as defined for
       the class PolicyGroup. See Section 13.1.1 for details.

The other two properties are defined in Reference [1] as follows:

```
NAME               CreationClassName
DESCRIPTION        This property is set to "CIM_PolicyRule", if the
                   PolicyRule object is directly instantiated.  Or,
                   it is equal to the class name of the PolicyRule
                   subclass that is instantiated.
SYNTAX             string [MaxLen 256]
QUALIFIER          key


NAME               PolicyRuleName
DESCRIPTION        The identifying name of this policy rule.
SYNTAX             string [MaxLen 256]
QUALIFIER          key
```

13.2. Naming Instances of PolicyCondition and Its Subclasses

The CIM keys of the PolicyCondition class are:

   o SystemCreationClassName
   o SystemName
   o PolicyRuleCreationClassName
   o PolicyRuleName
   o CreationClassName
   o PolicyConditionName

Note that none of the keys are defined as propagated, although they
appear to fit this convention. The reason for this difference is
because (as indicated in Sections 5.1 and 6.4) the PolicyCondition
class is used to represent both reusable and rule-specific
conditions. This, in turn, affects what associations are valid for
an instance of PolicyCondition, and how that instance is named.

In an ideal world, an instance of the PolicyCondition class would be
scoped either by its PolicyRepository (for a reusable condition) or
by its PolicyRule (for a rule-specific condition). However, CIM has
the restriction that a given class can only be "weak" to one other
class (i.e., defined by one weak association).

To work within the restrictions of CIM naming, it is necessary to
"simulate" weak associations between PolicyCondition and PolicyRule,
and between PolicyCondition and PolicyRepository, through a technique
we'll call manual key propagation. Strictly speaking, manual key
propagation isn't key propagation at all. But it has the same effect
as (true) key propagation, so the name fits.

Figure 9 illustrates how manual propagation works in the case of
PolicyCondition. (Note that only the key properties are shown for
each of the classes.) In the figure, the line composed of 'I's
indicates class inheritance, the one composed of 'P's indicates
(true) key propagation via the weak aggregation PolicyRuleInSystem,
and the ones composed of 'M's indicate manual key propagation.

```
        +------------------+
        |      System      |
        +------------------+
        |CreationClassName |
        |Name              |
        +------------------+
                 ^       P
                 I       PPPPPPPPPPPPPPPPPPPPPPPPPPPPP
                 I                               P
        +------------------+        +---------------v--------------+
        |    AdminDomain   |        |          PolicyRule          |
        +------------------+        +------------------------------+
        |CreationClassName |        | System.CreationClassName     |
        |Name              |        | System.Name                  |
        +------------------+        | CreationClassName            |
                 ^                  | PolicyRuleName               |
                 I                  +------------------------------+
                 I                               M
                 I                               M
        +------------------+                     M
        | PolicyRepository |                     M
        +------------------+                     M
        |CreationClassName |                     M
        |Name              |                     M
        +------------------+                     M
                 M                               M
                 M                               M
                 M                               M
              +----v-------------------v----+
              |        PolicyCondition      |
              +-----------------------------+
              | SystemCreationClassName     |
              | SystemName                  |
              | PolicyRuleCreationClassName |
              | PolicyRuleName              |
              | CreationClassName           |
              | PolicyConditionName         |
              +-----------------------------+
```

       Figure 9. Manual Key Propagation for Naming PolicyConditions

Looking at Figure 9, we see that two key properties,
CreationClassName and Name, are defined in the System class, and
inherited by its subclasses AdminDomain and PolicyRepository. Since
PolicyRule is weak to System, these two keys are propagated to it; it
also has its own keys CreationClassName and PolicyRuleName.

A similar approach, though not automatic, is used in "manual key
propagation". Here is the approach for rule-specific and reusable
PolicyConditions:

o The manual propagation of keys from PolicyRule to PolicyCondition
   involves copying the values of PolicyRule's four key properties
   into four similarly named key properties in PolicyCondition. From
   the point of view of the CIM specification language, the property
   SystemName in PolicyCondition is a completely new key property.
   However, the relationship to the Name property in System is
   defined in the description of SystemName.

o The manual propagation of keys from PolicyRepository to
   PolicyCondition works in exactly the same way for the first two
   key properties. However, since PolicyRepository doesn't include
   PolicyRule properties, the PolicyRuleCreationClassName and
   PolicyRuleName have no values. A special value, "No Rule", is
   assigned to both of these properties in this case, indicating that
   this instance of PolicyCondition is not named within the scope of
   any particular policy rule.

The following section defines the specific CIM keys for
PolicyCondition.

13.2.1. PolicyCondition's CIM Keys

PolicyCondition's key properties are defined in Reference [1] as
follows:

NAME   SystemCreationClassName
DESCRIPTION   SystemCreationClassName represents the class
                name of the CIM System object providing the
                naming scope for the instance of PolicyCondition.
                For a rule-specific policy condition, this is the
                type of system (e.g., the name of the class that
                created this instance) in whose context the policy
                rule is defined. For a reusable policy condition,
                this is set to "CIM_PolicyRepository", if the
                PolicyRepository object is directly instantiated.
                Or, it is equal to the class name of the
                PolicyRepository subclass that is instantiated.
SYNTAX   string [MaxLen 256]

```
QUALIFIER          key

NAME               SystemName
DESCRIPTION        The name of the System object in whose scope this
                   policy condition is defined.  This property
                   completes the identification of the System object.
                   For a rule-specific policy condition, this is the
                   name of the instance of the system in whose
                   context the policy rule is defined.  For a
                   reusable policy condition, this is name of the
                   instance of PolicyRepository that holds the policy
                   condition.
SYNTAX             string [MaxLen 256]
QUALIFIER          key

NAME               PolicyRuleCreationClassName
DESCRIPTION        For a rule-specific policy condition, this
                   property identifies the class name of the policy
                   rule instance, in whose scope this instance of
                   PolicyCondition exists.  For a reusable policy
                   condition, this property is set to a special
                   value, "No Rule", indicating that this instance
                   of PolicyCondition is not unique to one policy
                   rule.
SYNTAX             string [MaxLen 256]
QUALIFIER          key

NAME               PolicyRuleName
DESCRIPTION        For a rule-specific policy condition,
                   PolicyRuleName completes the identification of
                   the PolicyRule object with which this condition
                   is associated.  For a reusable policy condition,
                   a special value, "No Rule", is used to indicate
                   that this condition is reusable.
SYNTAX             string [MaxLen 256]
QUALIFIER          key

NAME               CreationClassName
DESCRIPTION        The class name of the PolicyCondition subclass
                   that is instantiated.
SYNTAX             string [MaxLen 256]
QUALIFIER          key

NAME               PolicyConditionName
DESCRIPTION        The identifying name of this policy condition.
SYNTAX             string [MaxLen 256]
QUALIFIER          key
```

13.3. Naming Instances of PolicyAction and Its Subclasses

   From the point of view of naming, the PolicyAction class and its
   subclasses work exactly like the PolicyCondition class and its
   subclasses. See Section 13.2 and 13.2.1 for details.

   Specifically, the CIM keys of PolicyAction are:

      o SystemCreationClassName
      o SystemName
      o PolicyRuleCreationClassName
      o PolicyRuleName
      o CreationClassName
      o PolicyActionName

   They are defined in Reference [1] as follows:

   NAME   SystemCreationClassName
   DESCRIPTION   SystemCreationClassName represents the class name
                     of the CIM System object providing the naming
                     scope for the instance of PolicyAction. For a
                     rule-specific policy action, this is the type of
                     system (e.g., the name of the class that created
                     this instance) in whose context the policy rule
                     is defined. For a reusable policy action, this
                     is set to "CIM_PolicyRepository", if the
                     PolicyRepository object is directly instantiated.
                     Or, it is equal to the class name of the
                     PolicyRepository subclass that is instantiated.
   SYNTAX   string [MaxLen 256]
   QUALIFIER   key

   NAME   SystemName
   DESCRIPTION   The name of the System object in whose scope this
                     policy action is defined. This property completes
                     the identification of the System object. For a
                     rule-specific policy action, this is the name of
                     the instance of the system in whose context the
                     policy rule is defined. For a reusable policy
                     action, this is name of the instance of
                     PolicyRepository that holds the policy action.
   SYNTAX   string [MaxLen 256]
   QUALIFIER   key

   NAME   PolicyRuleCreationClassName
   DESCRIPTION   For a rule-specific policy action, this property
                     identifies the class name of the policy rule
                     instance, in whose scope this instance of

```
                        PolicyAction exists.  For a reusable policy
                        action, this property is set to a special value,
                        "No Rule", indicating that this instance of
                        PolicyAction is not unique to one policy rule.
        SYNTAX          string [MaxLen 256]
        QUALIFIER       key


        NAME            PolicyRuleName
        DESCRIPTION     For a rule-specific policy action, PolicyRuleName
                        completes the identification of the PolicyRule
                        object with which this action is associated.  For
                        a reusable policy action, a special value, "No
                        Rule", is used to indicate that this action is
                        reusable.
        SYNTAX          string [MaxLen 256]
        QUALIFIER       key


        NAME            CreationClassName
        DESCRIPTION     The class name of the PolicyAction subclass that is
                        instantiated.
        SYNTAX          string [MaxLen 256]
        QUALIFIER       key


        NAME            PolicyActionName
        DESCRIPTION     The identifying name of this policy action.
        SYNTAX          string [MaxLen 256]
        QUALIFIER       key
```

13.4. Naming Instances of PolicyRepository

   An instance of PolicyRepository is named by the two key properties
   CreationClassName and Name that it inherits from its superclass
   AdminDomain. These properties are actually defined in AdminDomain's
   superclass, System, and then inherited by AdminDomain.

   For instances of PolicyRepository itself, the value of
   CreationClassName must be "CIM_PolicyRepository". (Recall that for
   readability the prefix "CIM_" has been omitted from all class names
   in this document). If a subclass of PolicyRepository (perhaps
   QosPolicyRepository) is defined and instantiated, then the class name
   "CIM_QosPolicyRepository" is used in CreationClassName.

   The Name property simply completes the identification of the instance
   of PolicyRepository.

13.5. Role of the CreationClassName Property in Naming

   To provide for more flexibility in instance naming, CIM makes use of
   a property called CreationClassName. The idea of CreationClassName
   is to provide another dimension that can be used to avoid naming
   collisions, in the specific case of instances belonging to two
   different subclasses of a common superclass. An example will
   illustrate how CreationClassName works.

   Suppose we have instances of two different subclasses of
   PolicyCondition, FrameRelayPolicyCondition and BgpPolicyCondition,
   and that these instances apply to the same context. If we had only
   the single key property PolicyConditionName available for
   distinguishing the two instances, then a collision would result from
   naming both of the instances with the key value PCName = "PC-1".
   Thus policy administrators from widely different disciplines would
   have to coordinate their naming of PolicyConditions for this context.

   With CreationClassName, collisions of this type can be eliminated,
   without requiring coordination among the policy administrators. The
   two instances can be distinguished by giving their CreationClassNames
   different values. One instance is now identified with the two keys

   CreationClassName = "FrameRelayPolicyCondition" + PCName = "PC-1",

   while the other is identified with

   CreationClassName = "BgpPolicyCondition" + PCName = "PC-1".

   Each of the instantiable classes in the Core Model includes the
   CreationClassName property as a key in addition to its own class-
   specific key property.

13.6. Object References

   Today, all CIM associations involve two object references. CIM
   decomposes an object reference into two parts: a high-order part
   that identifies an object manager and namespace, and a model path
   that identifies an object instance within a namespace. The model
   path, in turn, can be decomposed into an object class identifier and
   a set of key values needed to identify an instance of that class.

   Because the object class identifier is part of the model path, a CIM
   object reference is strongly typed. The GroupComponent object
   reference in the PolicyGroupInPolicyGroup association, for example,
   can only point to an instance of PolicyGroup, or to an instance of a

subclass of PolicyGroup. Contrast this with LDAP, where a DN pointer
is completely untyped: it identifies (by DN) an entry, but places no
restriction on that entry's object class(es).

An important difference between CIM property definitions and LDAP
attribute type definitions was identified earlier in Section 6:
while an LDAP attribute type definition has global scope, a CIM
property definition applies only to the class in which it is defined.
Thus properties having the same name in two different classes are
free to have different data types. CIM takes advantage of this
flexibility by allowing the data type of an object reference to be
overridden in a subclass of the association class in which it was
initially defined.

For example, the object reference GroupComponent is defined in the
abstract aggregation class PolicyComponent to be a reference to an
instance of the class Policy. This data type for GroupComponent is
then overridden in subclasses of PolicyComponent. In
PolicyGroupInPolicyGroup, for example, GroupComponent becomes a
reference to an instance of PolicyGroup. But in
PolicyConditionInPolicyRule it becomes a reference to an instance of
PolicyRule. Of course there is not total freedom in this overriding
of object references. In order to remain consistent with its
abstract superclass, a subclass of PolicyComponent can only override
GroupComponent to be a reference to a subclass of Policy. A Policy
class is the generic context for the GroupComponent reference in
PolicyComponent.

14. Appendix B: The Core Policy MOF


```
// ======================================================================
// Title:     Core Policy MOF Specification 2.4
// Filename:  CIM_Policy24.MOF
// Version:   2.4
// Release:   0
// Description: The object classes below are listed in an order that
//              avoids forward references.  Required objects, defined
//         by other working groups, are omitted.
// Date: 06/27/2000
//     CIMCR516a - Rooted the model associations under Policy
//         Component or PolicyInSystem.  Corrected PolicyCondition/
//         PolicyActionInPolicyRepository to subclass from
//         PolicyInSystem (similar to Groups and Roles 'InSystem')
// ======================================================================
// Author:    DMTF SLA (Service Level Agreement) Working Group
// ======================================================================
// Pragmas
// ======================================================================
#pragma Locale ("en-US")



// ======================================================================
// Policy
// ======================================================================
   [Abstract, Description (
         "An abstract class describing common properties of all "
         "policy rule-related subclasses, such as PolicyGroup, Policy"
         "Rule and PolicyCondition. All instances of policy rule-"
         "related entities will be created from subclasses of CIM_"
         "Policy.  The exception to this statement is PolicyRepository "
         "which is a type of CIM_System.")
   ]
class CIM_Policy : CIM_ManagedElement
{
      [Description (
         "A user-friendly name of this policy-related object.")
      ]
   string CommonName;
      [Description (
         "An array of keywords for characterizing / categorizing "
         "policy objects.  Keywords are of one of two types: \n"
         "  o Keywords defined in this and other MOFs, or in DMTF "
         "    white papers.  These keywords provide a vendor-"
         "    independent, installation-independent way of "
         "    characterizing policy objects. \n"
         "  o Installation-dependent keywords for characterizing "
```

```
          "      policy objects.  Examples include 'Engineering', "
          "      'Billing', and 'Review in December 2000'.  \n"
          "This MOF defines the following keywords:  'UNKNOWN', "
          "'CONFIGURATION', 'USAGE', 'SECURITY', 'SERVICE', "
          "'MOTIVATIONAL', 'INSTALLATION', and 'EVENT'.  These "
          "concepts are self-explanatory and are further discussed "
          "in the SLA/Policy White Paper.  One additional keyword "
          "is defined: 'POLICY'.  The role of this keyword is to "
          "identify policy-related instances that may not be otherwise "
          "identifiable, in some implementations.  The keyword 'POLICY' "
          "is NOT mutually exclusive of the other keywords "
          "specified above.")
       ]
    string PolicyKeywords [];
};

// ===================================================================
//    PolicyComponent
// ===================================================================
    [Association, Abstract, Aggregation, Description (
          "CIM_PolicyComponent is a generic association used to "
          "establish 'part of' relationships between the subclasses of "
          "CIM_Policy.  For example, the PolicyConditionInPolicyRule "
          "association defines that PolicyConditions are part of a "
          "PolicyRule.")
    ]
class CIM_PolicyComponent
{
        [Aggregate, Key, Description (
          "The parent Policy in the association.")
         ]
     CIM_Policy REF GroupComponent;
        [Key, Description (
          "The child/part Policy in the association.")
         ]
     CIM_Policy REF PartComponent;
};

// ===================================================================
//    PolicyInSystem
// ===================================================================
    [Association, Abstract, Description (
          "  CIM_PolicyInSystem is a generic association used to "
          "establish dependency relationships between Policies and the "
          "Systems that host them.  These Systems may be ComputerSystems "
          "where Policies are 'running' or they may be Policy"
          "Repositories where Policies are stored.  This relationship "
          "is similar to the concept of CIM_Services being dependent "
```

```
           "on CIM_Systems as defined by the HostedService "
           "association.  \n"
           "  Cardinality is Max(1) for the Antecedent/System "
           "reference since Policies can only be hosted in at most one "
           "System context.  Some subclasses of the association will "
           "further refine this definition to make the Policies Weak "
           "to Systems.  Other subclasses of PolicyInSystem will "
           "define an optional hosting relationship.  Examples of each "
           "of these are the PolicyRuleInSystem and PolicyConditionIn"
           "PolicyRepository associations, respectively.")
    ]
class CIM_PolicyInSystem : CIM_Dependency
{
        [Override ("Antecedent"), Max (1), Description (
           "The hosting System.")
        ]
     CIM_System REF Antecedent;
        [Override ("Dependent"), Description (
           "The hosted Policy.")
        ]
     CIM_Policy REF Dependent;
};

// ====================================================================
// PolicyGroup
// ====================================================================
    [Description (
           "A container for either a set of related PolicyGroups "
           "or a set of related PolicyRules, but not both.  Policy"
           "Groups are defined and named relative to the CIM_System "
           "which provides their context.")
    ]
class CIM_PolicyGroup : CIM_Policy
{
        [Propagated("CIM_System.CreationClassName"),
           Key, MaxLen (256),
           Description ("The scoping System's CreationClassName.")
        ]
    string SystemCreationClassName;
        [Propagated("CIM_System.Name"),
           Key, MaxLen (256),
           Description ("The scoping System's Name.")
        ]
    string SystemName;
        [Key, MaxLen (256), Description (
           "CreationClassName indicates the name of the class or the "
           "subclass used in the creation of an instance.  When used "
           "with the other key properties of this class, this property "
```

```
            "allows all instances of this class and its subclasses to "
            "be uniquely identified.") ]
    string CreationClassName;
        [Key, MaxLen (256), Description (
            "A user-friendly name of this PolicyGroup.")
        ]
    string PolicyGroupName;
};


// ===================================================================
//    PolicyGroupInPolicyGroup
// ===================================================================
    [Association, Aggregation, Description (
            "A relationship that aggregates one or more lower-level "
            "PolicyGroups into a higher-level Group.  A Policy"
            "Group may aggregate either PolicyRules or other Policy"
            "Groups, but not both.")
    ]
class CIM_PolicyGroupInPolicyGroup : CIM_PolicyComponent
{
        [Override ("GroupComponent"), Aggregate, Description (
         "A PolicyGroup that aggregates other Groups.")
        ]
    CIM_PolicyGroup REF GroupComponent;
        [Override ("PartComponent"), Description (
         "A PolicyGroup aggregated by another Group.")
        ]
    CIM_PolicyGroup REF PartComponent;
};


// ===================================================================
//    PolicyGroupInSystem
// ===================================================================
    [Association, Description (
            "An association that links a PolicyGroup to the System "
            "in whose scope the Group is defined.")
    ]
class CIM_PolicyGroupInSystem : CIM_PolicyInSystem
{
        [Override ("Antecedent"), Min(1), Max(1), Description (
         "The System in whose scope a PolicyGroup is defined.")
        ]
    CIM_System REF Antecedent;
        [Override ("Dependent"), Weak, Description (
         "A PolicyGroup named within the scope of a System.")
        ]
    CIM_PolicyGroup REF Dependent;
};
```

```
// ====================================================================
// PolicyRule
// ====================================================================
   [Description (
          "  The central class for representing the 'If Condition then "
          "Action' semantics associated with a policy rule. "
          "A PolicyRule condition, in the most general sense, is "
          "represented as either an ORed set of ANDed conditions "
          "(Disjunctive Normal Form, or DNF) or an ANDed set of ORed "
          "conditions (Conjunctive Normal Form, or CNF). Individual "
          "conditions may either be negated (NOT C) or unnegated (C). "
          "The actions specified by a PolicyRule are to be performed "
          "if and only if the PolicyRule condition (whether it is "
          "represented in DNF or CNF) evaluates to TRUE.\n\n"
          "   "
          "The conditions and actions associated with a PolicyRule "
          "are modeled, respectively, with subclasses of Policy"
          "Condition and PolicyAction.  These condition and action "
          "objects are tied to instances of PolicyRule by the Policy"
          "ConditionInPolicyRule and PolicyActionInPolicyRule "
          "aggregations.\n\n"
          "   "
          "A PolicyRule may also be associated with one or more policy "
          "time periods, indicating the schedule according to which the "
          "policy rule is active and inactive.  In this case it is the "
          "PolicyRuleValidityPeriod aggregation that provides this "
          "linkage.\n\n"
          "   "
          "The PolicyRule class uses the property ConditionListType, to "
          "indicate whether the conditions for the rule are in DNF or "
          "CNF.  The PolicyConditionInPolicyRule aggregation contains "
          "two additional properties to complete the representation of "
          "the Rule's conditional expression.  The first of these "
          "properties is an integer to partition the referenced "
          "PolicyConditions into one or more groups, and the second is a "
          "Boolean to indicate whether a referenced Condition is "
          "negated.  An example shows how ConditionListType and these "
          "two additional properties provide a unique representation "
          "of a set of PolicyConditions in either DNF or CNF.\n\n"
          "   "
          "Suppose we have a PolicyRule that aggregates five "
          "PolicyConditions C1  through C5, with the following values "
          "in the properties of the five PolicyConditionInPolicyRule "
          "associations:\n"
          "     C1:  GroupNumber = 1, ConditionNegated = FALSE\n "
          "     C2:  GroupNumber = 1, ConditionNegated = TRUE\n   "
          "     C3:  GroupNumber = 1, ConditionNegated = FALSE\n "
          "     C4:  GroupNumber = 2, ConditionNegated = FALSE\n "
```

```
             "     C5:  GroupNumber = 2, ConditionNegated = FALSE\n\n "
          "   "
          "If ConditionListType = DNF, then the overall condition for "
          "the PolicyRule is:\n"
          "        (C1 AND (NOT C2) AND C3) OR (C4 AND C5)\n\n"
          "   "
          "On the other hand, if ConditionListType = CNF, then the "
          "overall condition for the PolicyRule is:\n"
          "        (C1 OR (NOT C2) OR C3) AND (C4 OR C5)\n\n"
          "   "
          "In both cases, there is an unambiguous specification of "
          "the overall condition that is tested to determine whether "
          "to perform the PolicyActions associated with the PolicyRule.")
    ]
class CIM_PolicyRule : CIM_Policy
{
        [Propagated("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.")
        ]
    string SystemCreationClassName;
        [Propagated("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.")
        ]
    string SystemName;
        [Key, MaxLen (256), Description (
            "CreationClassName indicates the name of the class or the "
            "subclass used in the creation of an instance.  When used "
            "with the other key properties of this class, this property "
            "allows all instances of this class and its subclasses to "
            "be uniquely identified.") ]
    string CreationClassName;
        [Key, MaxLen (256), Description (
            "A user-friendly name of this PolicyRule.")
        ]
    string PolicyRuleName;
        [Description (
            "Indicates whether this PolicyRule is administratively "
            "enabled, administratively disabled, or enabled for "
            "debug.  When the property has the value 3 (\"enabledFor"
            "Debug\"), the entity evaluating the PolicyConditions is "
            "instructed to evaluate the conditions for the Rule, but not "
            "to perform the actions if the PolicyConditions evaluate to "
            "TRUE.  This serves as a debug vehicle when attempting to "
            "determine what policies would execute in a particular "
            "scenario, without taking any actions to change state "
            "during the debugging.  The default value is 1
```

```
(\"enabled\")."),
          ValueMap { "1", "2", "3" },
          Values { "enabled", "disabled", "enabledForDebug" }
          ]
     uint16 Enabled;
         [Description (
            "Indicates whether the list of PolicyConditions "
            "associated with this PolicyRule is in disjunctive "
            "normal form (DNF) or conjunctive normal form (CNF)."
            "The default value is 1 (\"DNF\")."),
          ValueMap { "1", "2" },
          Values { "DNF", "CNF" }
          ]
     uint16 ConditionListType;
         [Description (
            "A free-form string that can be used to provide "
            "guidelines on how this PolicyRule should be used.")
          ]
     string RuleUsage;
         [Description (
            "A non-negative integer for prioritizing this Policy"
            "Rule relative to other Rules.  A larger value "
            "indicates a higher priority.  The default value is 0.")
          ]
     uint16 Priority;
         [Description (
            "A flag indicating that the evaluation of the Policy"
            "Conditions and execution of PolicyActions (if the "
            "Conditions evaluate to TRUE) is required.  The "
            "evaluation of a PolicyRule MUST be attempted if the "
            "Mandatory property value is TRUE.  If the Mandatory "
            "property is FALSE, then the evaluation of the Rule "
            "is 'best effort' and MAY be ignored.")
          ]
     boolean Mandatory;
         [Description (
            "This property gives a policy administrator a way "
            "of specifying how the ordering of the PolicyActions "
            "associated with this PolicyRule is to be interpreted. "
            "Three values are supported:\n"
            "  o mandatory(1): Do the actions in the indicated "
            "     order, or don't do them at all.\n"
            "  o recommended(2): Do the actions in the indicated "
            "     order if you can, but if you can't do them in this "
            "     order, do them in another order if you can.\n"
            "  o dontCare(3): Do them -- I don't care about the "
            "     order.\n"
            "The default value is 3 (\"dontCare\")."),
```

```
             ValueMap { "1", "2", "3" },
             Values { "mandatory", "recommended", "dontCare" }
             ]
       uint16 SequencedActions;
            [Description (
             "This property represents the roles and role combinations "
             "associated with a PolicyRule.  Each value represents one "
             "role or role combination.  Since this is a multi-valued "
             "property, more than one role or combination can be associated "
             "with a single policy rule.  Each value is a string of the "
             "form:\n"
             "   <RoleName>[&&<RoleName>]*\n"
             "where the individual role names appear in alphabetical order "
             "(according to the collating sequence for UCS-2).")
             ]
       string PolicyRoles [];
   };


   // ===================================================================
   //    PolicyRuleInPolicyGroup
   // ===================================================================
      [Association, Aggregation, Description (
            "A relationship that aggregates one or more PolicyRules "
            "into a PolicyGroup.  A PolicyGroup may aggregate either "
            "PolicyRules or other PolicyGroups, but not both.")
      ]
   class CIM_PolicyRuleInPolicyGroup : CIM_PolicyComponent
   {
            [Override ("GroupComponent"), Aggregate, Description (
             "A PolicyGroup that aggregates one or more PolicyRules.")
             ]
       CIM_PolicyGroup REF GroupComponent;
            [Override ("PartComponent"), Description (
             "A PolicyRule aggregated by a PolicyGroup.")
             ]
       CIM_PolicyRule REF PartComponent;
   };


   // ===================================================================
   //    PolicyRuleInSystem
   // ===================================================================
      [Association, Description (
            "An association that links a PolicyRule to the System "
            "in whose scope the Rule is defined.")
      ]
   class CIM_PolicyRuleInSystem : CIM_PolicyInSystem
   {
            [Override ("Antecedent"), Min(1), Max(1), Description (
```

```
          "The System in whose scope a PolicyRule is defined.")
          ]
      CIM_System REF Antecedent;
          [Override ("Dependent"), Weak, Description (
           "A PolicyRule named within the scope of a System.")
           ]
      CIM_PolicyRule REF Dependent;
};


// ===================================================================
// PolicyRepository
// ===================================================================
   [Description (
          "A class representing an administratively defined "
          "container for reusable policy-related information. "
          "This class does not introduce any additional "
          "properties beyond those in its superclass "
          "AdminDomain.  It does, however, participate in a "
          "number of unique associations."
          "\n\n"
          "An instance of this class uses the NameFormat value"
          "\"PolicyRepository\", which is defined in the AdminDomain"
          "class.")
   ]
class CIM_PolicyRepository : CIM_AdminDomain
{
};


// ===================================================================
//    PolicyRepositoryInPolicyRepository
// ===================================================================
   [Association, Aggregation, Description (
          "A relationship that aggregates one or more lower-level "
          "PolicyRepositories into a higher-level Repository.")
   ]
class CIM_PolicyRepositoryInPolicyRepository : CIM_SystemComponent
{
          [Override ("GroupComponent"), Aggregate, Description (
           "A PolicyRepository that aggregates other Repositories.")
           ]
      CIM_PolicyRepository REF GroupComponent;
          [Override ("PartComponent"), Description (
           "A PolicyRepository aggregated by another Repository.")
           ]
      CIM_PolicyRepository REF PartComponent;
};


// ===================================================================
```

```
// PolicyCondition
// ===================================================================
    [Abstract, Description (
          "A class representing a rule-specific or reusable policy "
          "condition to be evaluated in conjunction with a Policy"
          "Rule.  Since all operational details of a PolicyCondition "
          "are provided in subclasses of this object, this class is "
          "abstract.")
    ]
class CIM_PolicyCondition : CIM_Policy
{
        [Key, MaxLen (256), Description (
           "  The name of the class or the subclass used in the "
          "creation of the System object in whose scope this "
          "PolicyCondition is defined.\n\n"
          "  "
          "This property helps to identify the System object in "
          "whose scope this instance of PolicyCondition exists. "
          "For a rule-specific PolicyCondition, this is the System "
          "in whose context the PolicyRule is defined.  For a "
          "reusable PolicyCondition, this is the instance of "
          "PolicyRepository (which is a subclass of System) that "
          "holds the Condition.\n\n"
          "  "
          "Note that this property, and the analogous property "
          "SystemName, do not represent propagated keys from an "
          "instance of the class System.  Instead, they are "
          "properties defined in the context of this class, which "
          "repeat the values from the instance of System to which "
          "this PolicyCondition is related, either directly via the "
          "PolicyConditionInPolicyRepository aggregation or indirectly "
          "via the PolicyConditionInPolicyRule aggregation.")
         ]
    string SystemCreationClassName;
        [Key, MaxLen (256), Description (
          "  The name of the System object in whose scope this "
          "PolicyCondition is defined.\n\n"
          "  "
          "This property completes the identification of the System "
          "object in whose scope this instance of PolicyCondition "
          "exists.  For a rule-specific PolicyCondition, this is the "
          "System in whose context the PolicyRule is defined.  For a "
          "reusable PolicyCondition, this is the instance of "
          "PolicyRepository (which is a subclass of System) that "
          "holds the Condition.")
         ]
    string SystemName;
        [Key, MaxLen (256), Description (
```

```
            "For a rule-specific PolicyCondition, the "
            "CreationClassName of the PolicyRule object with which "
            "this Condition is associated.  For a reusable Policy"
            "Condition, a special value, 'NO RULE', should be used to "
            "indicate that this Condition is reusable and not "
            "associated with a single PolicyRule.")
          ]
    string PolicyRuleCreationClassName;
        [Key, MaxLen (256), Description (
         "For a rule-specific PolicyCondition, the name of "
         "the PolicyRule object with which this Condition is "
         "associated.  For a reusable PolicyCondition, a "
         "special value, 'NO RULE', should be used to indicate "
         "that this Condition is reusable and not associated "
         "with a single PolicyRule.")
          ]
    string PolicyRuleName;
        [Key, MaxLen (256), Description (
            "CreationClassName indicates the name of the class or the "
            "subclass used in the creation of an instance.  When used "
            "with the other key properties of this class, this property "
            "allows all instances of this class and its subclasses to "
            "be uniquely identified.") ]
    string CreationClassName;
        [Key, MaxLen (256), Description (
            "A user-friendly name of this PolicyCondition.")
          ]
    string PolicyConditionName;
};


// ====================================================================
//    PolicyConditionInPolicyRule
// ====================================================================
    [Association, Aggregation, Description (
         "  A PolicyRule aggregates zero or more instances of the "
         "PolicyCondition class, via the PolicyConditionInPolicyRule "
         "association.  A Rule that aggregates zero Conditions is not "
         "valid -- it may, however, be in the process of being entered "
         "into a PolicyRepository or being defined for a System.  Note "
         "that a PolicyRule should have no effect until it is valid.\n\n"
         "   "
         "The Conditions aggregated by a PolicyRule are grouped into "
         "two levels of lists: either an ORed set of ANDed sets of "
         "conditions (DNF, the default) or an ANDed set of ORed sets "
         "of conditions (CNF).  Individual PolicyConditions in these "
         "lists may be negated.  The property ConditionListType "
         "specifies which of these two grouping schemes applies to a "
         "particular PolicyRule.\n\n"
```

```
          "   "
          "In either case, PolicyConditions are used to determine whether "
          "to perform the PolicyActions associated with the
PolicyRule.\n\n"
          "   "
          "One or more PolicyTimePeriodConditions may be among the "
          "conditions associated with a PolicyRule via the Policy"
          "ConditionInPolicyRule association.  In this case, the time "
          "periods are simply additional Conditions to be evaluated "
          "along with any others that are specified for the Rule. ")
    ]
class CIM_PolicyConditionInPolicyRule : CIM_PolicyComponent
{
        [Override ("GroupComponent"), Aggregate, Description (
         "This property represents the PolicyRule that "
         "contains one or more PolicyConditions.")
         ]
    CIM_PolicyRule REF GroupComponent;
        [Override ("PartComponent"), Description (
         "This property holds the name of a PolicyCondition "
         "contained by one or more PolicyRules.")
         ]
    CIM_PolicyCondition REF PartComponent;
        [Description (
         "Unsigned integer indicating the group to which the "
         "PolicyCondition identified by the ContainedCondition "
         "property belongs.  This integer segments the Conditions "
         "into the ANDed sets (when the ConditionListType is "
         "\"DNF\") or similarly the ORed sets (when the Condition"
         "ListType is \"CNF\") that are then evaluated.")
         ]
    uint16 GroupNumber;
        [Description (
         "Indication of whether the Condition identified by "
         "the ContainedCondition property is negated.  TRUE "
         "indicates that the PolicyCondition IS negated, FALSE "
         "indicates that it IS NOT negated.")
         ]
    boolean ConditionNegated;
};

// ====================================================================
//    PolicyConditionInPolicyRepository
// ====================================================================
    [Association, Description (
         "  A class representing the hosting of reusable "
         "PolicyConditions by a PolicyRepository.  A reusable Policy"
         "Condition is always related to a single PolicyRepository, "
```

```
            "via this aggregation.\n\n"
            "   "
            "Note, that an instance of PolicyCondition can be either "
            "reusable or rule-specific.  When the Condition is rule-"
            "specific, it shall not be related to any "
            "PolicyRepository via the PolicyConditionInPolicyRepository "
            "aggregation.")
    ]
class CIM_PolicyConditionInPolicyRepository : CIM_PolicyInSystem
{
        [Override ("Antecedent"), Max(1), Description (
         "This property identifies a PolicyRepository "
         "hosting one or more PolicyConditions.  A reusable "
         "PolicyCondition is always related to exactly one "
         "PolicyRepository via the PolicyConditionInPolicyRepository "
         "aggregation.  The [0..1] cardinality for this property "
         "covers the two types of PolicyConditions:  0 for a "
         "rule-specific PolicyCondition, 1 for a reusable one.")
         ]
     CIM_PolicyRepository REF Antecedent;
        [Override ("Dependent"), Description (
         "This property holds the name of a PolicyCondition"
         "hosted in the PolicyRepository. ")
         ]
     CIM_PolicyCondition REF Dependent;
};


// ===================================================================
// PolicyTimePeriodCondition
// ===================================================================
    [Description (
         "  This class provides a means of representing the time "
         "periods during which a PolicyRule is valid, i.e., active. "
         "At all times that fall outside these time periods, the "
         "PolicyRule has no effect.  A Rule is treated as valid "
         "at ALL times, if it does not specify a "
         "PolicyTimePeriodCondition.\n\n"
         "   "
         "In some cases a Policy Consumer may need to perform "
         "certain setup / cleanup actions when a PolicyRule becomes "
         "active / inactive.  For example, sessions that were "
         "established while a Rule was active might need to "
         "be taken down when the Rule becomes inactive.  In other "
         "cases, however, such sessions might be left up.  In this "
         "case, the effect of deactivating the PolicyRule would "
         "just be to prevent the establishment of new sessions. \n\n"
         "   "
         "Setup / cleanup behaviors on validity period "
```

```
        "transitions are not currently addressed by the Policy "
        "Model, and must be specified in 'guideline' documents or "
        "via subclasses of CIM_PolicyRule, CIM_PolicyTimePeriod"
        "Condition or other concrete subclasses of CIM_Policy.  If "
        "such behaviors need to be under the control of the policy "
        "administrator, then a mechanism to allow this control "
        "must also be specified in the subclasses.\n\n"
        "   "
        "PolicyTimePeriodCondition is defined as a subclass of "
        "PolicyCondition.  This is to allow the inclusion of "
        "time-based criteria in the AND/OR condition definitions "
        "for a PolicyRule.\n\n"
        "   "
        "Instances of this class may have up to five properties "
        "identifying time periods at different levels.  The values "
        "of all the properties present in an instance are ANDed "
        "together to determine the validity period(s) for the "
        "instance.  For example, an instance with an overall "
        "validity range of January 1, 2000 through December 31, "
        "2000; a month mask that selects March and April; a "
        "day-of-the-week mask that selects Fridays; and a time "
        "of day range of 0800 through 1600 would be represented "
        "using the following time periods:\n"
        "   Friday, March  5, 2000, from 0800 through 1600;\n "
        "   Friday, March 12, 2000, from 0800 through 1600;\n "
        "   Friday, March 19, 2000, from 0800 through 1600;\n "
        "   Friday, March 26, 2000, from 0800 through 1600;\n "
        "   Friday, April  2, 2000, from 0800 through 1600;\n "
        "   Friday, April  9, 2000, from 0800 through 1600;\n "
        "   Friday, April 16, 2000, from 0800 through 1600;\n "
        "   Friday, April 23, 2000, from 0800 through 1600;\n "
        "   Friday, April 30, 2000, from 0800 through 1600.\n\n"
        "   "
        "Properties not present in an instance of "
        "PolicyTimePeriodCondition are implicitly treated as having "
        "their value 'always enabled'.  Thus, in the example above, "
        "the day-of-the-month mask is not present, and so the "
        "validity period for the instance implicitly includes a "
        "day-of-the-month mask that selects all days of the month. "
        "If this 'missing property' rule is applied to its fullest, we "
        "see that there is a second way to indicate that a Policy"
        "Rule is always enabled: associate with it an instance of "
        "PolicyTimePeriodCondition whose only properties with "
        "specific values are its key properties.")
   ]
class CIM_PolicyTimePeriodCondition : CIM_PolicyCondition
{
        [Description (
```

                  " This property identifies an overall range of calendar "
                 "dates and times over which a PolicyRule is valid. It is "
                 "formatted as a string representing a start date and time, "
                 "in which the character 'T' indicates the beginning of the "
                 "time portion, followed by the solidus character '/', "
                 "followed by a similar string representing an end date and "
                 "time. The first date indicates the beginning of the range, "
                 "while the second date indicates the end. Thus, the second "
                 "date and time must be later than the first. Date/times are "
                 "expressed as substrings of the form yyyymmddThhmmss. For "
                 "example: \n"
                 "   20000101T080000/20000131T120000 defines \n"
                 "   January 1, 2000, 0800 through January 31, 2000, noon\n\n"
                 " "
                 "There are also two special cases in which one of the "
                 "date/time strings is replaced with a special string defined "
                 "in RFC 2445.\n "
                 "   o If the first date/time is replaced with the string "
                 "   'THISANDPRIOR', then the property indicates that a "
                 "   PolicyRule is valid [from now] until the date/time "
                 "   that appears after the '/'.\n"
                 "   o If the second date/time is replaced with the string "
                 "   'THISANDFUTURE', then the property indicates that a "
                 "   PolicyRule becomes valid on the date/time that "
                 "   appears before the '/', and remains valid from that "
                 "   point on. "),
              ModelCorrespondence {
                 "CIM_PolicyTimePeriodCondition.MonthOfYearMask",
                 "CIM_PolicyTimePeriodCondition.DayOfMonthMask",
                 "CIM_PolicyTimePeriodCondition.DayOfWeekMask",
                 "CIM_PolicyTimePeriodCondition.TimeOfDayMask",
                 "CIM_PolicyTimePeriodCondition.LocalOrUtcTime"}
              ]
         string TimePeriod;
              [Octetstring, Description (
                 " The purpose of this property is to refine the valid time "
                 "period that is defined by the TimePeriod property, by "
                 "explicitly specifying in which months the PolicyRule is "
                 "valid. These properties work together, with the "
                 "TimePeriod used to specify the overall time period in "
                 "which the PolicyRule is valid, and the MonthOfYearMask used "
                 "to pick out the months during which the Rule is valid.\n\n"
                 " "
                 "This property is formatted as an octet string, structured "
                 "as follows:\n"
                 "   o a 4-octet length field, indicating the length of the "
                 "   entire octet string; this field is always set to "
                 "   0x00000006 for this property;\n"

```
            "    o a 2-octet field consisting of 12 bits identifying the "
            "       12 months of the year, beginning with January and "
            "       ending with December, followed by 4 bits that are "
            "       always set to '0'.  For each month, the value '1' "
            "       indicates that the policy is valid for that month, "
            "       and the value '0' indicates that it is not valid.\n\n"
            "    "
            "The value 0x000000060830, for example, indicates that a "
            "PolicyRule is valid only in the months May, November, "
            "and December.\n\n"
            "    "
            "If a value for this property is not provided, then the "
            "PolicyRule is treated as valid for all twelve months, and "
            "only restricted by its TimePeriod property value and the "
            "other Mask properties."),
        ModelCorrespondence {
        "CIM_PolicyTimePeriodCondition.TimePeriod",
        "CIM_PolicyTimePeriodCondition.LocalOrUtcTime"}
        ]
    uint8 MonthOfYearMask[];
        [Octetstring, Description (
         "  The purpose of this property is to refine the valid time "
         "period that is defined by the TimePeriod property, by "
         "explicitly specifying in which days of the month the Policy"
         "Rule is valid.  These properties work together, "
         "with the TimePeriod used to specify the overall time period "
         "in which the PolicyRule is valid, and the DayOfMonthMask used "
         "to pick out the days of the month during which the Rule "
         "is valid.\n\n "
         "    "
         "This property is formatted as an octet string, structured "
         "as follows:\n"
         "    o a 4-octet length field, indicating the length of the "
         "       entire octet string; this field is always set to "
         "       0x0000000C for this property; \n"
         "    o an 8-octet field consisting of 31 bits identifying "
         "       the days of the month counting from the beginning, "
         "       followed by 31 more bits identifying the days of the "
         "       month counting from the end, followed by 2 bits that "
         "       are always set to '0'.  For each day, the value '1' "
         "       indicates that the policy is valid for that day, and "
         "       the value '0' indicates that it is not valid. \n\n"
         "    "
         "The value 0x0000000C8000000100000000, for example, "
         "indicates that a PolicyRule is valid on the first and "
         "last days of the month.\n\n "
         "    "
         "For months with fewer than 31 days, the digits corresponding "
```

```
         "to days that the months do not have (counting in both "
         "directions) are ignored.\n\n"
         "  "
         "If a value for this property is not provided, then the "
         "PolicyRule is treated as valid for all days of the month, and "
         "only restricted by its TimePeriod property value and the "
         "other Mask properties."),
      ModelCorrespondence {
      "CIM_PolicyTimePeriodCondition.TimePeriod",
      "CIM_PolicyTimePeriodCondition.LocalOrUtcTime"}
      ]
   uint8 DayOfMonthMask[];
      [Octetstring, Description (
       "  The purpose of this property is to refine the valid time "
       "period that is defined by the TimePeriod property, by "
       "explicitly specifying in which days of the month the Policy"
       "Rule is valid.  These properties work together, "
       "with the TimePeriod used to specify the overall time period "
       "in which the PolicyRule is valid, and the DayOfWeekMask used "
       "to pick out the days of the week during which the Rule "
       "is valid.\n\n "
       "  "
       "This property is formatted as an octet string, structured "
       "as follows:\n "
       "  o a 4-octet length field, indicating the length of the "
       "    entire octet string; this field is always set to "
       "    0x00000005 for this property;\n"
       "  o a 1-octet field consisting of 7 bits identifying the 7 "
       "    days of the week, beginning with Sunday and ending with "
       "    Saturday, followed by 1 bit that is always set to '0'. "
       "    For each day of the week, the value '1' indicates that "
       "    the policy is valid for that day, and the value '0' "
       "    indicates that it is not valid. \n\n"
       "  "
       "The value 0x000000057C, for example, indicates that a "
       "PolicyRule is valid Monday through Friday.\n\n"
       "  "
       "If a value for this property is not provided, then the "
       "PolicyRule is treated as valid for all days of the week, "
       "and only restricted by its TimePeriod property value and "
       "the other Mask properties."),
      ModelCorrespondence {
      "CIM_PolicyTimePeriodCondition.TimePeriod",
      "CIM_PolicyTimePeriodCondition.LocalOrUtcTime"}
      ]
   uint8 DayOfWeekMask[];
      [Description (
       "  The purpose of this property is to refine the valid time "
```

          "period that is defined by the TimePeriod property, by "
          "explicitly specifying a range of times in a day during which "
          "the PolicyRule is valid. These properties work "
          "together, with the TimePeriod used to specify the overall "
          "time period in which the PolicyRule is valid, and the "
          "TimeOfDayMask used to pick out the range of time periods "
          "in a given day of during which the Rule is valid. \n\n"
          "   "
          "This property is formatted in the style of RFC 2445: a "
          "time string beginning with the character 'T', followed by "
          "the solidus character '/', followed by a second time string. "
          "The first time indicates the beginning of the range, while "
          "the second time indicates the end. Times are expressed as "
          "substrings of the form 'Thhmmss'. \n\n"
          "   "
          "The second substring always identifies a later time than "
          "the first substring. To allow for ranges that span "
          "midnight, however, the value of the second string may be "
          "smaller than the value of the first substring. Thus, "
          "'T080000/T210000' identifies the range from 0800 until 2100, "
          "while 'T210000/T080000' identifies the range from 2100 until "
          "0800 of the following day. \n\n"
          "   "
          "When a range spans midnight, it by definition includes "
          "parts of two successive days. When one of these days is "
          "also selected by either the MonthOfYearMask, "
          "DayOfMonthMask, and/or DayOfWeekMask, but the other day is "
          "not, then the policy is active only during the portion of "
          "the range that falls on the selected day. For example, if "
          "the range extends from 2100 until 0800, and the day of "
          "week mask selects Monday and Tuesday, then the policy is "
          "active during the following three intervals:\n"
          "   From midnight Sunday until 0800 Monday; \n"
          "   From 2100 Monday until 0800 Tuesday; \n"
          "   From 2100 Tuesday until 23:59:59 Tuesday. \n\n"
          "   "
          "If a value for this property is not provided, then the "
          "PolicyRule is treated as valid for all hours of the day, "
          "and only restricted by its TimePeriod property value and "
          "the other Mask properties."),
        ModelCorrespondence {
        "CIM_PolicyTimePeriodCondition.TimePeriod",
        "CIM_PolicyTimePeriodCondition.LocalOrUtcTime"}
        ]
    string TimeOfDayMask;
        [Description (
         " This property indicates whether the times represented "
         "in the TimePeriod property and in the various Mask "

```
              "properties represent local times or UTC times.  There is "
              "no provision for mixing of local times and UTC times:  the "
              "value of this property applies to all of the other "
              "time-related properties."),
           ValueMap { "1", "2" },
           Values { "localTime", "utcTime" },
           ModelCorrespondence {
           "CIM_PolicyTimePeriodCondition.TimePeriod",
           "CIM_PolicyTimePeriodCondition.MonthOfYearMask",
           "CIM_PolicyTimePeriodCondition.DayOfMonthMask",
           "CIM_PolicyTimePeriodCondition.DayOfWeekMask",
           "CIM_PolicyTimePeriodCondition.TimeOfDayMask"}
           ]
     uint16 LocalOrUtcTime;
};


// =====================================================================
//    PolicyRuleValidityPeriod
// =====================================================================
    [Association, Aggregation, Description (
           "The PolicyRuleValidityPeriod aggregation represents "
           "scheduled activation and deactivation of a PolicyRule. "
           "If a PolicyRule is associated with multiple policy time "
           "periods via this association, then the Rule is active if "
           "at least one of the time periods indicates that it is "
           "active.  (In other words, the PolicyTimePeriodConditions "
           "are ORed to determine whether the Rule is active.)  A Time"
           "Period may be aggregated by multiple PolicyRules.  A Rule "
           "that does not point to a PolicyTimePeriodCondition via this "
           "association is, from the point of view of scheduling, "
           "always active.  It may, however, be inactive for other "
           "reasons.  For example, the Rule's Enabled property may "
           "be set to \"disabled\" (value=2).")
    ]
class CIM_PolicyRuleValidityPeriod : CIM_PolicyComponent
{
         [Override ("GroupComponent"), Aggregate, Description (
          "This property contains the name of a PolicyRule that "
          "contains one or more PolicyTimePeriodConditions.")
          ]
     CIM_PolicyRule REF GroupComponent;
         [Override ("PartComponent"), Description (
          "This property contains the name of a "
          "PolicyTimePeriodCondition defining the valid time periods "
          "for one or more PolicyRules.")
          ]
     CIM_PolicyTimePeriodCondition REF PartComponent;
};
```

```
   // ===================================================================
   // VendorPolicyCondition
   // ===================================================================
      [Description (
             "  A class that provides a general extension mechanism for "
             "representing PolicyConditions that have not been modeled "
             "with specific properties.  Instead, the two properties "
             "Constraint and ConstraintEncoding are used to define the "
             "content and format of the Condition, as explained below.\n\n"
             "  "
             "As its name suggests, VendorPolicyCondition is intended for "
             "vendor-specific extensions to the Policy Core Information "
             "Model.  Standardized extensions are not expected to use "
             "this class.")
      ]
   class CIM_VendorPolicyCondition : CIM_PolicyCondition
   {
             [Octetstring, Description (
              "This property provides a general extension mechanism for "
              "representing PolicyConditions that have not been "
              "modeled with specific properties.  The format of the "
              "octet strings in the array is left unspecified in "
              "this definition.  It is determined by the OID value "
              "stored in the property ConstraintEncoding.  Since "
              "ConstraintEncoding is single-valued, all the values of "
              "Constraint share the same format and semantics."),
              ModelCorrespondence {
                  "CIM_VendorPolicyCondition.ConstraintEncoding"}
             ]
       string Constraint [];
             [Description (
              "An OID encoded as a string, identifying the format "
              "and semantics for this instance's Constraint property."),
              ModelCorrespondence {
                  "CIM_VendorPolicyCondition.Constraint"}
             ]
       string ConstraintEncoding;
   };


   // ===================================================================
   // PolicyAction
   // ===================================================================
      [Abstract, Description (
             "A class representing a rule-specific or reusable policy "
             "action to be performed if the PolicyConditions for a Policy"
             "Rule evaluate to TRUE.  Since all operational details of a "
             "PolicyAction are provided in subclasses of this object, "
             "this class is abstract.")
```

```
     ]
class CIM_PolicyAction : CIM_Policy
{
        [Key, MaxLen (256), Description (
         "  The name of the class or the subclass used in the "
         "creation of the System object in whose scope this "
         "PolicyAction is defined. \n\n"
         "  "
         "This property helps to identify the System object in "
         "whose scope this instance of PolicyAction exists. "
         "For a rule-specific PolicyAction, this is the System "
         "in whose context the PolicyRule is defined.  For a "
         "reusable PolicyAction, this is the instance of "
         "PolicyRepository (which is a subclass of System) that "
         "holds the Action. \n\n"
         "  "
         "Note that this property, and the analogous property "
         "SystemName, do not represent propagated keys from an "
         "instance of the class System.  Instead, they are "
         "properties defined in the context of this class, which "
         "repeat the values from the instance of System to which "
         "this PolicyAction is related, either directly via the "
         "PolicyActionInPolicyRepository aggregation or indirectly "
         "via the PolicyActionInPolicyRule aggregation.")
        ]
     string SystemCreationClassName;
        [Key, MaxLen (256), Description (
         "  The name of the System object in whose scope this "
         "PolicyAction is defined. \n\n"
         "  "
         "This property completes the identification of the System "
         "object in whose scope this instance of PolicyAction "
         "exists.  For a rule-specific PolicyAction, this is the "
         "System in whose context the PolicyRule is defined.  For "
         "a reusable PolicyAction, this is the instance of "
         "PolicyRepository (which is a subclass of System) that "
         "holds the Action.")
        ]
     string SystemName;
        [Key, MaxLen (256), Description (
         "For a rule-specific PolicyAction, the CreationClassName "
         "of the PolicyRule object with which this Action is "
         "associated.  For a reusable PolicyAction, a "
         "special value, 'NO RULE', should be used to "
         "indicate that this Action is reusable and not "
         "associated with a single PolicyRule.")
        ]
     string PolicyRuleCreationClassName;
```

```
         [Key, MaxLen (256), Description (
          "For a rule-specific PolicyAction, the name of "
          "the PolicyRule object with which this Action is "
          "associated.  For a reusable PolicyAction, a "
          "special value, 'NO RULE', should be used to "
          "indicate that this Action is reusable and not "
          "associated with a single PolicyRule.")
          ]
     string PolicyRuleName;
         [Key, MaxLen (256), Description (
            "CreationClassName indicates the name of the class or the "
            "subclass used in the creation of an instance.  When used "
            "with the other key properties of this class, this property "
            "allows all instances of this class and its subclasses to "
            "be uniquely identified.") ]
     string CreationClassName;
         [Key, MaxLen (256), Description (
          "A user-friendly name of this PolicyAction.")
          ]
     string PolicyActionName;
};


// ===================================================================
//    PolicyActionInPolicyRepository
// ===================================================================
   [Association, Description (
          "  A class representing the hosting of reusable "
          "PolicyActions by a PolicyRepository.  A reusable Policy"
          "Action is always related to a single PolicyRepository, "
          "via this aggregation.\n\n"
          "   "
          "Note, that an instance of PolicyAction can be either "
          "reusable or rule-specific.  When the Action is rule-"
          "specific, it shall not be related to any "
          "PolicyRepository via the PolicyActionInPolicyRepository "
          "aggregation.")
   ]
class CIM_PolicyActionInPolicyRepository : CIM_PolicyInSystem
{
         [Override ("Antecedent"), Max(1), Description (
          "This property represents a PolicyRepository "
          "hosting one or more PolicyActions.  A reusable "
          "PolicyAction is always related to exactly one "
          "PolicyRepository via the PolicyActionInPolicyRepository "
          "aggregation.  The [0..1] cardinality for this property "
          "covers the two types of PolicyActions:  0 for a "
          "rule-specific PolicyAction, 1 for a reusable one.")
          ]
```

```
    CIM_PolicyRepository REF Antecedent;
        [Override ("Dependent"), Description (
         "This property holds the name of a PolicyAction"
         "hosted in the PolicyRepository. ")
        ]
    CIM_PolicyAction REF Dependent;
};


// ===================================================================
//    PolicyActionInPolicyRule
// ===================================================================
   [Association, Aggregation, Description (
        "  A PolicyRule aggregates zero or more instances of the "
        "PolicyAction class, via the PolicyActionInPolicyRule "
        "association.  A Rule that aggregates zero Actions is not "
        "valid -- it may, however, be in the process of being entered "
        "into a PolicyRepository or being defined for a System. "
        "Alternately, the actions of the policy may be explicit in "
        "the definition of the PolicyRule.  Note that a PolicyRule "
        "should have no effect until it is valid.\n\n"
        "   "
        "The Actions associated with a PolicyRule may be given a "
        "required order, a recommended order, or no order at all.  For "
        "Actions represented as separate objects, the PolicyActionIn"
        "PolicyRule aggregation can be used to express an order. \n\n"
        "   "
        "This aggregation does not indicate whether a specified "
        "action order is required, recommended, or of no significance; "
        "the property SequencedActions in the aggregating instance of "
        "PolicyRule provides this indication.")
   ]
class CIM_PolicyActionInPolicyRule : CIM_PolicyComponent
{
        [Override ("GroupComponent"), Aggregate, Description (
         "This property represents the PolicyRule that "
         "contains one or more PolicyActions.")
        ]
    CIM_PolicyRule REF GroupComponent;
        [Override ("PartComponent"), Description (
         "This property holds the name of a PolicyAction "
         "contained by one or more PolicyRules.")
        ]
    CIM_PolicyAction REF PartComponent;
        [Description (
         "  This property provides an unsigned integer 'n' that"
         "indicates the relative position of a PolicyAction in the "
         "sequence of actions associated with a PolicyRule. "
         "When 'n' is a positive integer, it indicates a place "
```

```
            "in the sequence of actions to be performed, with "
            "smaller integers indicating earlier positions in the "
            "sequence.  The special value '0' indicates 'don't care'. "
            "If two or more PolicyActions have the same non-zero "
            "sequence number, they may be performed in any order, but "
            "they must all be performed at the appropriate place in the "
            "overall action sequence. \n\n"
            "   "
            "A series of examples will make ordering of PolicyActions "
            "clearer: \n"
            "   o If all actions have the same sequence number, "
            "     regardless of whether it is '0' or non-zero, any "
            "     order is acceptable.\n "
            "   o The values: \n"
            "        1:ACTION A \n"
            "        2:ACTION B \n"
            "        1:ACTION C \n"
            "        3:ACTION D \n"
            "     indicate two acceptable orders: A,C,B,D or C,A,B,D, "
            "     since A and C can be performed in either order, but "
            "     only at the '1' position. \n"
            "   o The values: \n"
            "        0:ACTION A \n"
            "        2:ACTION B \n"
            "        3:ACTION C \n"
            "        3:ACTION D \n"
            "     require that B,C, and D occur either as B,C,D or as "
            "     B,D,C.  Action A may appear at any point relative to "
            "     B, C, and D.  Thus the complete set of acceptable "
            "     orders is:  A,B,C,D; B,A,C,D; B,C,A,D; B,C,D,A; "
            "     A,B,D,C; B,A,D,C; B,D,A,C; B,D,C,A. \n\n"
            "   "
            "Note that the non-zero sequence numbers need not start "
            "with '1', and they need not be consecutive.  All that "
            "matters is their relative magnitude.")
          ]
     uint16 ActionOrder;
};


// ===================================================================
// VendorPolicyAction
// ===================================================================
    [Description (
            "  A class that provides a general extension mechanism for "
            "representing PolicyActions that have not been modeled "
            "with specific properties.  Instead, the two properties "
            "ActionData and ActionEncoding are used to define the "
            "content and format of the Action, as explained below.\n\n"
```

```
             "   "
             "As its name suggests, VendorPolicyAction is intended for "
             "vendor-specific extensions to the Policy Core Information "
             "Model.  Standardized extensions are not expected to use "
             "this class.")  ]
class CIM_VendorPolicyAction : CIM_PolicyAction
{
        [Octetstring, Description (
         "This property provides a general extension mechanism for "
         "representing PolicyActions that have not been "
         "modeled with specific properties.  The format of the "
         "octet strings in the array is left unspecified in "
         "this definition.  It is determined by the OID value "
         "stored in the property ActionEncoding.  Since "
         "ActionEncoding is single-valued, all the values of "
         "ActionData share the same format and semantics."),
         ModelCorrespondence {
            "CIM_VendorPolicyAction.ActionEncoding"}
        ]
    string ActionData [];
        [Description (
         "An OID encoded as a string, identifying the format "
         "and semantics for this instance's ActionData property."),
         ModelCorrespondence {
            "CIM_VendorPolicyAction.ActionData"}
        ]
    string ActionEncoding;
};


// ===================================================================
// end of file
// ===================================================================
```

15. Full Copyright Statement

Acknowledgement