I n t e r n a t i o n a l   T e l e c o m m u n i c a t i o n   U n i o n

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.1036
(11/2007)

SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY

Telecommunication security

# Framework for creation, storage, distribution and enforcement of policies for network security

ITU-T Recommendation X.1036

ITU-T X-SERIES RECOMMENDATIONS

**DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY**

| | |
|---|---|
| PUBLIC DATA NETWORKS | |
| Services and facilities | X.1–X.19 |
| Interfaces | X.20–X.49 |
| Transmission, signalling and switching | X.50–X.89 |
| Network aspects | X.90–X.149 |
| Maintenance | X.150–X.179 |
| Administrative arrangements | X.180–X.199 |
| OPEN SYSTEMS INTERCONNECTION | |
| Model and notation | X.200–X.209 |
| Service definitions | X.210–X.219 |
| Connection-mode protocol specifications | X.220–X.229 |
| Connectionless-mode protocol specifications | X.230–X.239 |
| PICS proformas | X.240–X.259 |
| Protocol Identification | X.260–X.269 |
| Security Protocols | X.270–X.279 |
| Layer Managed Objects | X.280–X.289 |
| Conformance testing | X.290–X.299 |
| INTERWORKING BETWEEN NETWORKS | |
| General | X.300–X.349 |
| Satellite data transmission systems | X.350–X.369 |
| IP-based networks | X.370–X.379 |
| MESSAGE HANDLING SYSTEMS | X.400–X.499 |
| DIRECTORY | X.500–X.599 |
| OSI NETWORKING AND SYSTEM ASPECTS | |
| Networking | X.600–X.629 |
| Efficiency | X.630–X.639 |
| Quality of service | X.640–X.649 |
| Naming, Addressing and Registration | X.650–X.679 |
| Abstract Syntax Notation One (ASN.1) | X.680–X.699 |
| OSI MANAGEMENT | |
| Systems Management framework and architecture | X.700–X.709 |
| Management Communication Service and Protocol | X.710–X.719 |
| Structure of Management Information | X.720–X.729 |
| Management functions and ODMA functions | X.730–X.799 |
| SECURITY | X.800–X.849 |
| OSI APPLICATIONS | |
| Commitment, Concurrency and Recovery | X.850–X.859 |
| Transaction processing | X.860–X.879 |
| Remote operations | X.880–X.889 |
| Generic applications of ASN.1 | X.890–X.899 |
| OPEN DISTRIBUTED PROCESSING | X.900–X.999 |
| **TELECOMMUNICATION SECURITY** | **X.1000–** |

*For further details, please refer to the list of ITU-T Recommendations.*

## ITU-T Recommendation X.1036

# Framework for creation, storage, distribution and enforcement of policies for network security

**Summary**

ITU-T Recommendation X.1036 establishes a set of network security policies that will drive the security controls of a system or a service. It also specifies the framework for the creation, storage, distribution and enforcement of policies for network security that can be applied to various network conditions and devices.

**Source**

ITU-T Recommendation X.1036 was approved on 13 November 2007 by ITU-T Study Group 17 (2005-2008) under the ITU-T Recommendation A.8 procedure.

## CONTENTS

# ITU-T Recommendation X.1036

# Framework for creation, storage, distribution and enforcement of policies for network security

## 1 Scope

Most of current security research has been devoted to the development of security systems such as IDS (intrusion detection system) or firewall. Note, however, that those systems that are currently available are not generally interoperable because each system has its own special functionality and control mechanism. This proves to be greatly bothersome to operators who have to control one or more networks including several security systems.

As such, controlling different security systems effectively or easily managing them in a unified manner has become a hot issue. This Recommendation defines the network security policy information model (NSPIM) extended on the basis of policy core information model (PCIM) and PCIM extension (PCIMe) of IETF in representing several policies used in the policy-based network security system. NSPIM is highlighted as a solution for effectively controlling various network devices. NSPIM delivers consistent, unified, and an understandable view of a network without the implementation details. Such benefit of NSPIM is enhanced as the network grows increasingly complex and offers more services. To model the interaction between network elements, services, and clients of the network, the flexible and extensible information model for security policy is required for the design of each component of NSPIM. Management, signalling/control, and user planes as defined in [ITU-T X.805] can be used to determine the necessary controls needed to support a security policy.

The scope of this Recommendation covers the establishment of a set of security policies for protecting the network and proposal of a network security policy framework that creates, stores, distributes and executes these network security policies. The network security policy information model is also introduced to represent the security policies used for intrusion detection and response in several modules in the network security system.

## 2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

| | |
|---|---|
| [ITU-T X.800] | ITU-T Recommendation X.800 (1991), *Security architecture for Open Systems Interconnection for CCITT applications*. |
| [ITU-T X.805] | ITU-T Recommendation X.805 (2003), *Security architecture for systems providing end-to-end communications*. |
| [ITU-T X.810] | ITU-T Recommendation X.810 (1995) \| ISO/IEC 10181-1:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Overview*. |
| [ITU-T X.812] | ITU-T Recommendation X.812 (1995) \| ISO/IEC 10181-3:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework.* |

| | |
|---|---|
| [IETF RFC 3060] | IETF RFC 3060 (2001), *Policy Core Information Model – Version 1 Specification*, <http://www.ietf.org/rfc/rfc3060.txt> |
| [IEFT RFC 3460] | IETF RFC 3460 (2003), *Policy Core Information Model (PCIM) Extensions*, <http://www.ietf.org/rfc/rfc3460.txt> |

## 3 Terms and definitions

### 3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

**3.1.1 associations**: [IETF RFC 3060]

**3.1.2 aggregations**: [IETF RFC 3060]

### 3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

**3.2.1 action**: This operation is governed by a policy rule. Any action evaluated to be TRUE based on a rule is triggered.

**3.2.2 attack**: An attack may either be a known attack or an unknown attack. The known attack means that the pattern or packet for attack is opened. Although the pattern or packet for attack is not opened for the unknown attack, it refers to the behaviour related to worsening network situation.

**3.2.3 condition**: This is an expression of a condition type; its values are used to specify a constraint within a policy rule. A given condition can be negated using the NOT operator.

**3.2.4 information model**: This is an abstraction and representation of the entities in a managed environment as well as their properties, attributes and operations, and way of relating to each other. It is independent of any specific repository, application, protocol or platform.

**3.2.5 policy**: A policy defines one or more rules. Each rule binds one or more actions to sets of conditions describing by whom (users), for what (systems, applications), and under what circumstances (time, day of the week, date) the actions may be triggered.

**3.2.6 policy conflict**: It defines the actions of two rules contradicting each other. The entity implementing the policy will not be able to determine which action to perform. To prevent this situation, the implementers of policy systems must provide conflict detection and avoidance or resolution mechanisms.

**3.2.7 role**: This is a string characterizing a particular function of a network element or an interface that can be used to identify particular behaviours associated with the element. It is a selector for policy rules to determine the applicability of the rule to a particular network element. Roles abstract the capabilities and/or use of network devices and resources.

**3.2.8 rule**: This is a policy component that binds an action to the conditions governing whether or not the action is performed. When controlling network resources, the action is usually intended to provide a service. The following is a simplified expression for a rule: IF condition, THEN action.

## 4 Abreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

| | |
|---|---|
| CIM | Common Information Model |
| COPS | Common Open Policy Service |
| DMTF | Distributed Management Task Force |

| | |
|---|---|
| ICMP | Internet Control Message Protocol |
| IDS | Intrusion Detection System |
| LDAP | Lightweight Directory Access Protocol |
| NSPIM | Network Security Policy Information Model |
| PBNM | Policy-Based Network Management |
| PCIM | Policy Core Information Model |
| PCIMe | PCIM extension |
| PDU | Policy Decision Unit |
| PEP | Policy Enforcement Point |
| PES | Policy Enforcement System |
| PIB | Policy Information Base |
| PMU | Policy Management Unit |
| PR | Policy Repository |

## 5 Conventions

None.

## 6 Network security policy

The network service and system can be protected by monitoring known attacks blocking harmful packets and limiting traffic congestion. Security policies play an important role in enabling intrusion detection and automated responses. Depending on the role, the network security policy is divided into 5 sub-policies, each of which is expressed as condition/action pairs.

This can be generated by administrators or security managers and enforced by the security agents. The condition attribute and action classified according to the types of policies are shown in Table 1.

**Table 1 – Network security policies**

| Name | Condition attribute | Action |
|---|---|---|
| Attack Detection Policy | Packet Header Fields, Packet Payload Fields | Alert<br>Drop |
| Access Control Policy | Source IP Address, Destination IP Address, Source Port, Destination Port, Protocol, Direction, TCP 6 Flags, ICMP Type, ICMP Code | Permit<br>Deny |
| Alert Control Policy | Source IP Address, Destination IP Address, Source Port, Destination Port, Protocol, Attack ID | Filtering<br>Sampling |
| Traffic Control Policy | Source IP Address, Destination IP Address, Source Port, Destination Port, Protocol, Direction | Forward<br>Rate Limit |
| Routing Control Policy | Destination IP Address | Drop |

Attack Detection Policy can be used to detect and mitigate the known attacks. The access control policy, traffic control policy and routing control policy can be used to detect and mitigate the unknown attacks.

### 6.1 Attack detection policy

In detecting known attacks, the attack detection policy can refer to the attack detection rules from intrusion detection systems. The monitoring device drops a packet and/or sends an alert to the policy server after comparing the packet with the pattern defined in the attack detection rule.

### 6.2 Access control policy

The access control policy decides on permission or denial for incoming packets to a firewall or a router in accordance with the value of packet header fields.

### 6.3 Alert control policy

The alert control policy controls the transmission of alerts from IDS. This policy allows a policy enforcement system to send the sample of alerts or to filter alerts.

### 6.4 Traffic control policy

The traffic control policy prescribes the control against excessive traffic in a router or a traffic control device. This policy limits network traffic and helps increase network performance. In other words, it provides a basic level of security for network access.

### 6.5 Routing control policy

The routing control policy forwards a packet to a router's bit bucket for manipulating bad traffic. In other words, it discards unwanted packets. This policy works only on the destination address since it is actually part of the forwarding logic.

## 7 Network security policy information model

The network security policy information model (NSPIM) referred to as a common information model is constructed based on policy core information model (PCIM, [IETF RFC 3060]) and PCIM extension (PCIMe, [IETF RFC 3460]) of IETF in representing the policies used in the policy-based network security system and schema of a policy repository based on NSPIM. PCIM presents the object-oriented information model for representing policy information under joint development in the IETF policy framework working group and as extensions to the common information model (CIM) activity in the distributed management task force (DMTF). NSPIM defines two hierarchies of object classes: structural classes representing policy information and control of policies, and association classes indicating how instances of the structural classes are related to each other. Properly designed and implemented security mechanisms (dimensions), as defined in [ITU-T X.805], can be used to support security policy and facilitate the classes and rules set by security management.

### 7.1 Class definitions of NSPIM

The classes of CIM and PCIM/PCIMe are not covered by the scope of this Recommendation. Therefore, only newly defined classes in NSPIM are described.

#### 7.1.1 Class "AlertControlRule"

| | |
|---|---|
| NAME | AlertControlRule |
| DESCRIPTION | The class is defined for the associated condition and action for alert control |
| DERIVED FROM | PolicyRule |
| PROPERTIES | (none) |

### 7.1.2 Class "PacketFilteringRule"

| | |
|---|---|
| NAME | PacketFilteringRule |
| DESCRIPTION | The class is defined when dropping or permitting a specified packet or session |
| DERIVED FROM | PolicyRule |
| PROPERTIES | FilteringPriority |

#### 7.1.2.1 Property "FilteringPriority"

| | |
|---|---|
| NAME | FilteringPriority |
| DESCRIPTION | This indicates whether to drop or permit a specified packet or session |
| SYNTAX | uint16 |

### 7.1.3 Class "TrafficControlRule"

| | |
|---|---|
| NAME | TrafficControlRule |
| DESCRIPTION | The class is defined for the control of network traffic congestion |
| DERIVED FROM | PolicyRule |
| PROPERTIES | (none) |

### 7.1.4 Class "AttackDetectionRule"

| | |
|---|---|
| NAME | AttackDetectionRule |
| DESCRIPTION | The class is defined for Attack Detection |
| DERIVED FROM | PolicyRule |
| PROPERTIES | IntrusionImpact, Msg, Reference, AttackName, SignatureID, Revision, Priority, ClassType |

#### 7.1.4.1 Property "IntrusionImpact"

| | |
|---|---|
| NAME | IntrusionImpact |
| DESCRIPTION | This is an enumeration indicating the effect of intrusion |
| SYNTAX | uint16 |
| VALUES | Very Large (1), Large (2), Middle (3), Small (4), Very Small (5) |

#### 7.1.4.2 Property "Msg"

| | |
|---|---|
| NAME | Msg |
| DESCRIPTION | This property is used to describe the contents of detection rules |
| SYNTAX | string |

#### 7.1.4.3 Property "Reference"

| | |
|---|---|
| NAME | Reference |
| DESCRIPTION | This property presents the system referred to in detection rules |
| SYNTAX | string |

#### 7.1.4.4 Property "AttackName"

| | |
|---|---|
| NAME | AttackName |
| DESCRIPTION | This property presents the name of the attack |
| SYNTAX | string |

#### 7.1.4.5 Property "SignatureID"

| | |
|---|---|
| NAME | SignatureID |
| DESCRIPTION | This property is used for an identifier of the attack signature |
| SYNTAX | uint16 |

#### 7.1.4.6 Property "Revision"

| | |
|---|---|
| NAME | Revision |
| DESCRIPTION | This property is used to identify the revision of the rules |
| SYNTAX | uint32 |

#### 7.1.4.7 Property "Priority"

| | |
|---|---|
| NAME | Priority |
| DESCRIPTION | This property is used to present the priority of the rules |
| SYNTAX | uint32 |

#### 7.1.4.8 Property "ClassType"

| | |
|---|---|
| NAME | ClassType |
| DESCRIPTION | This property presents the category where the rule belongs |
| SYNTAX | uint32 |

### 7.1.5 Class "RoutingControlRule"

| | |
|---|---|
| NAME | RoutingControlRule |
| DESCRIPTION | The class is defined for the routing control |
| DERIVED FROM | PolicyRule |
| PROPERTIES | (none) |

### 7.1.6 Class "AlertControlCondition"

| | |
|---|---|
| NAME | AlertControlCondition |
| DESCRIPTION | The class is defined for filtering alerts. The AlertControlCondition class instance consists of combinations of the source IP address, destination IP address, source port, destination port, protocol and attack ID. |
| DERIVED FROM | CompoundPolicyCondition |
| PROPERTIES | (none) |

### 7.1.7 Class "PacketFilteringCondition"

| | |
|---|---|
| NAME | PacketFilteringCondition |
| DESCRIPTION | The class represents the condition of an access control policy for special packets or sessions. The PacketFilteringCondition class instance consists of combinations of the source IP address, destination IP address, source port, destination port, protocol, direction, TCP flags, ICMP type and ICMP code. |
| DERIVED FROM | CompoundPolicyCondition |
| PROPERTIES | (none) |

### 7.1.8 Class "TrafficControlCondition"

| | |
|---|---|
| NAME | TrafficControlCondition |
| DESCRIPTION | The class is defined for congested traffic control. The TrafficControlCondition class instance consists of combinations of the source IP address, destination IP address, source port, destination port and protocol. |
| DERIVED FROM | CompoundPolicyCondition |
| PROPERTIES | (none) |

### 7.1.9 Class "RoutingControlCondition"

| | |
|---|---|
| NAME | RoutingControlCondition |
| DESCRIPTION | The class is defined as a condition of routing control rule. The RoutingControlCondition class instance consists of the destination IP address. |
| DERIVED FROM | CompoundPolicyCondition |
| PROPERTIES | (none) |

### 7.1.10 Class "PacketMonitoringCondition"

| | |
|---|---|
| NAME | PacketMonitoringCondition |
| DESCRIPTION | The class is defined for the condition of AttackDetectionRule. This PacketMonitoringCondition consists of DetailedPacketFilter and/or ExtendedHeaderFilter through FilterList association. |
| DERIVED FROM | PolicyCondition |
| PROPERTIES | (none) |

### 7.1.11 Abstract Class "AlertControlAction"

| | |
|---|---|
| NAME | AlertControlAction |
| DESCRIPTION | The class is an abstract class for alert transmission |
| DERIVED FROM | PolicyAction |
| PROPERTIES | (none) |

### 7.1.12 Class "AlertSamplingAction"

| | |
|---|---|
| NAME | AlertSamplingAction |
| DESCRIPTION | The class represents an action that aggregates the same alerts and transmits the alert |
| DERIVED FROM | AlertControlAction |
| PROPERTIES | TimeInterval |

#### 7.1.12.1 Property "TimeInterval"

| | |
|---|---|
| NAME | TimeInterval |
| DESCRIPTION | This property presents the collection time of the associated information |
| SYNTAX | uint16 |

### 7.1.13 Class "AlertFilteringAction"

| | |
|---|---|
| NAME | AlertFilteringAction |
| DESCRIPTION | The class is defined for the alert filtering action |
| DERIVED FROM | AlertControlAction |
| PROPERTIES | (none) |

### 7.1.14 Class "PacketFilteringAction"

| | |
|---|---|
| NAME | PacketFilteringAction |
| DESCRIPTION | This class is defined when representing for packet filtering action |
| DERIVED FROM | PolicyAction |
| PROPERTIES | AccessStatus |

#### 7.1.14.1 Property "AccessStatus"

| | |
|---|---|
| NAME | AccessStatus |
| DESCRIPTION | This indicates whether to block or permit a specified packet or session |
| SYNTAX | uint16 |
| VALUES | 1[Permit], 2[Block] |
| DEFAULT VALUE | 2[Block] |

### 7.1.15 Class "TrafficControlAction"

| | |
|---|---|
| NAME | TrafficControlAction |
| DESCRIPTION | This class is defined for traffic control |
| DERIVED FROM | PolicyAction |
| PROPERTIES | AverageRate, PacketRate, NormalBucket, ExtendedBucket, ConformAction, ExceedAction, AccessControlAction |

#### 7.1.15.1 Property "AverageRate"

| | |
|---|---|
| NAME | AverageRate |
| DESCRIPTION | This is a non-negative integer for the average number of bits per second |
| SYNTAX | uint16 |
| DEFAULT VALUE | 0 |

#### 7.1.15.2 Property "PacketRate"

| | |
|---|---|
| NAME | PacketRate |
| DESCRIPTION | This is a non-negative integer for the average number of packets per second |
| SYNTAX | uint16 |
| DEFAULT VALUE | 0 |

#### 7.1.15.3 Property "NormalBucket"

| | |
|---|---|
| NAME | NormalBucket |
| DESCRIPTION | This is a non-negative integer for the size of a normal bucket |
| SYNTAX | uint16 |
| DEFAULT VALUE | 0 |

#### 7.1.15.4 Property "ExtendedBucket"

| | |
|---|---|
| NAME | ExtendedBucket |
| DESCRIPTION | This is a non-negative integer for the size of an extended bucket |
| SYNTAX | uint16 |
| DEFAULT VALUE | 0 |

#### 7.1.15.5 Property "ConformAction"

| | |
|---|---|
| NAME | ConformAction |
| DESCRIPTION | This property presents the action for conformity. It passes or drops the excess size of the normal bucket |
| SYNTAX | uint16 |
| VALUES | 1[Pass], 2[Drop] |
| DEFAULT VALUE | 1[Pass] |

#### 7.1.15.6 Property "ExceedAction"

| | |
|---|---|
| NAME | ExceedAction |
| DESCRIPTION | This property presents the action for exceed. It drops the excess size of the extended bucket |
| SYNTAX | uint16 |
| DEFAULT VALUE | 1[Drop] |

### 7.1.16 Class "BlackHoleAction"

| | |
|---|---|
| NAME | BlackHoleAction |
| DESCRIPTION | The class is defined for the action of routing control |
| DERIVED FROM | PolicyAction |
| PROPERTIES | (none) |

### 7.1.17 Class "ResponseAction"

| | |
|---|---|
| NAME | ResponseAction |
| DESCRIPTION | The class is defined for representing response action against attack detection |
| DERIVED FROM | PolicyAction |
| PROPERTIES | AlertSending, PacketHandling |

#### 7.1.17.1 Property "AlertSending"

| | |
|---|---|
| NAME | AlertSending |
| DESCRIPTION | This is a flag indicating whether or not to send alert information |
| SYNTAX | Boolean |

#### 7.1.17.2 Property "PacketHandling"

| | |
|---|---|
| NAME | PacketHandling |
| DESCRIPTION | This is an enumeration indicating the decision that the packet will be ignored, dropped, or terminated by sending the FIN packet |
| SYNTAX | uint16 |
| VALUES | 1[NotApplicable], 2[Drop], 3[Terminate] |
| DEFAULT VALUE | 1[NotApplicable] |

### 7.1.18 Class "PolicyAttackIDVariable"

| | |
|---|---|
| NAME | PolicyAttackIDVariable |
| DESCRIPTION | The class is the attack ID for the AlertControlCondition class |
| DERIVED FROM | PolicyExplicitVariable |
| PROPERTIES | (none) |

### 7.1.19 Class "PolicyTCPUrgVariable"

| | |
|---|---|
| NAME | PolicyTCPUrgVariable |
| DESCRIPTION | The class represents the URG field of the TCP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.20 Class "PolicyTCPAckVariable"

| | |
|---|---|
| NAME | PolicyTCPAckVariable |
| DESCRIPTION | The class represents the ACK field of the TCP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.21 Class "PolicyTCPPshVariable"

| | |
|---|---|
| NAME | PolicyTCPPshVariable |
| DESCRIPTION | The class represents the PSH field of the TCP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.22 Class "PolicyTCPRstVariable"

| | |
|---|---|
| NAME | PolicyTCPRstVariable |
| DESCRIPTION | The class represents the RST field of the TCP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.23 Class "PolicyTCPSynVariable"

| | |
|---|---|
| NAME | PolicyTCPSynVariable |
| DESCRIPTION | The class represents the SYN field of the TCP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.24 Class "PolicyTCPFinVariable"

| | |
|---|---|
| NAME | PolicyTCPFinVariable |
| DESCRIPTION | The class represents the FIN field of the TCP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.25 Class "PolicyTCPRes1Variable"

| | |
|---|---|
| NAME | PolicyTCPRes1Variable |
| DESCRIPTION | The class represents the 5th reserved field of the TCP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.26 Class "PolicyTCPRes2Variable"

| | |
|---|---|
| NAME | PolicyTCPRes2Variable |
| DESCRIPTION | The class is the last reserved field of the TCP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.27 Class "PolicyTCPNonVariable"

| | |
|---|---|
| NAME | PolicyTCPNonVariable |
| DESCRIPTION | The class means that TCP flags should not be set |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.28 Class "PolicyICMPTypeVariable"

| | |
|---|---|
| NAME | PolicyICMPTypeVariable |
| DESCRIPTION | The class represents the type field of the ICMP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.29 Class "PolicyICMPCodeVariable"

| | |
|---|---|
| NAME | PolicyICMPCodeVariable |
| DESCRIPTION | The class represents the code field of the ICMP packet header |
| DERIVED FROM | PolicyImplicitVariable |
| PROPERTIES | (none) |

### 7.1.30 Class "DetailedPacketFilter"

| | |
|---|---|
| NAME | DetailedPacketFilter |
| DESCRIPTION | The class is defined when performing filtering on IP, TCP, UDP or ICMP headers |
| DERIVED FROM | FilterEntryBase |
| PROPERTIES | IPVersion, SrcIpAddress, SrcIpAddressMask, SrcIpAddressNegated, DstIpAddress, DstIpAddressMask, DstIpAddressNegated, SrcPortStart, SrcPortEnd, SrcPortNegated, DstPortStart, DstPortEnd, DstPortNegated, Protocol, IpFragOffset, IpFragOffsetFlag, IpFragOffsetNegated, IpTtl, IpTtlFlags, IpTos, IpTosNegated, IpId, IpOption, IpFragBit, IpFragBitMode, IpDsize, IpDsizeFlag, TcpFlag, TcpFlagMode, Flow, Flowbits, TcpSeq, TcpAck, Window, IcmpType, IcmpCode, IcmpId, IcmpSeq, Rpc, IpProto, IpProtoFlag, IpProtoNegated, Content, IpDataLength, IpDataLengthFlag, UdpDataLength, UdpChecksum, IcmpChecksum, CountThreshold, TimeThreshold |

#### 7.1.30.1 Property "IPVersion"

| | |
|---|---|
| NAME | IPVersion |
| DESCRIPTION | This property presents the version of the Internet protocol |
| SYNTAX | uint16 |
| VALUES | 4(IPv4), 6(IPv6) |

#### 7.1.30.2 Property "SrcIpAddress"

| | |
|---|---|
| NAME | SrcIpAddress |
| DESCRIPTION | This property presents the source of the IP address |
| SYNTAX | string |

#### 7.1.30.3 Property "SrcIpAddressMask"

| | |
|---|---|
| NAME | SrcIpAddressMask |
| DESCRIPTION | This property presents the value of the mask by which the source of the IP address is compared with the property of SrcIpAddress |
| SYNTAX | string |

#### 7.1.30.4 Property "SrcIpAddressNegated"

| | |
|---|---|
| NAME | SrcIpAddressNegated |
| DESCRIPTION | This is a flag indicating that the source of the IP address is negated |
| SYNTAX | boolean |

#### 7.1.30.5 Property "DstIpAddress"

| | |
|---|---|
| NAME | DstIpAddress |
| DESCRIPTION | This property presents the destination of the IP address |
| SYNTAX | string |

#### 7.1.30.6 Property "DstIpAddressMask"

| | |
|---|---|
| NAME | DstIpAddressMask |
| DESCRIPTION | This property presents the value of the mask by which the destination of the IP address is compared with the property of DstIpAddress |
| SYNTAX | string |

#### 7.1.30.7 Property "DstIpAddressNegated"

| | |
|---|---|
| NAME | DstIpAddressNegated |
| DESCRIPTION | This is a flag indicating that the destination of the IP address is negated |
| SYNTAX | boolean |

#### 7.1.30.8 Property "SrcPortStart"

| | |
|---|---|
| NAME | SrcPortStart |
| DESCRIPTION | This property presents the start number of the source port of UDP or TCP |
| SYNTAX | unit16 |

#### 7.1.30.9 Property "SrcPortEnd"

| | |
|---|---|
| NAME | SrcPortEnd |
| DESCRIPTION | This property presents the end number of the source port of UDP or TCP |
| SYNTAX | unit16 |

#### 7.1.30.10 Property "SrcPortNegated"

| | |
|---|---|
| NAME | SrcPortNegated |
| DESCRIPTION | This is a flag indicating that the source port is negated |
| SYNTAX | boolean |

#### 7.1.30.11 Property "DstPortStart"

| | |
|---|---|
| NAME | DstPortStart |
| DESCRIPTION | This property presents the start number of the destination port of UDP or TCP |
| SYNTAX | unit16 |

### 7.1.30.12 Property "DstPortEnd"

| | |
|---|---|
| NAME | DstPortEnd |
| DESCRIPTION | This property presents the end number of the destination port of UDP or TCP |
| SYNTAX | unit16 |

### 7.1.30.13 Property "DstPortNegated"

| | |
|---|---|
| NAME | DstPortNegated |
| DESCRIPTION | This is a flag indicating that the destination port is negated |
| SYNTAX | boolean |

### 7.1.30.14 Property "Protocol"

| | |
|---|---|
| NAME | Protocol |
| DESCRIPTION | This property presents the type of packet, such as IP, UDP, TCP and ICMP |
| SYNTAX | unit16 |
| VALUES | –1(IP), 6(TCP), 17(UDP), 1(ICMP) |

### 7.1.30.15 Property "IpFragOffset"

| | |
|---|---|
| NAME | IpFragOffset |
| DESCRIPTION | This property presents the Fragment Offset of the IP packet header |
| SYNTAX | unit16 |

### 7.1.30.16 Property "IpFragOffsetFlag"

| | |
|---|---|
| NAME | IpFragOffsetFlag |
| DESCRIPTION | This property presents a flag indicating the Fragment Offset of the IP packet header |
| SYNTAX | unit16 |
| VALUES | GT(1), LT(2) |

### 7.1.30.17 Property "IpFragOffsetNegated"

| | |
|---|---|
| NAME | IpFragOffsetNegated |
| DESCRIPTION | This is a flag indicating that the Fragment Offset value of the IP packet header is negated |
| SYNTAX | boolean |

#### 7.1.30.18 Property "IpTtl"

| | |
|---|---|
| NAME | IpTtl |
| DESCRIPTION | This property presents the value of the TTL field of the IPv4 packet header or the value of the Hop Limit field of the IPv6 packet header |
| SYNTAX | unit16 |

#### 7.1.30.19 Property "IpTtlFlags"

| | |
|---|---|
| NAME | IpTtlFlags |
| DESCRIPTION | This property presents a flag indicating TTL field of the IP packet header |
| SYNTAX | unit16 |
| VALUES | EQ(0), RG(1), LT(2), GT(3) |

#### 7.1.30.20 Property "IpTos"

| | |
|---|---|
| NAME | IpTos |
| DESCRIPTION | This property presents the value of TOS field of the IPv4 packet header or the value of Traffic class field of the IPv6 packet header |
| SYNTAX | unit16 |

#### 7.1.30.21 Property "IpTosNegated"

| | |
|---|---|
| NAME | IpTosNegated |
| DESCRIPTION | This is a flag indicating that the property of the IpTos is negated |
| SYNTAX | boolean |

#### 7.1.30.22 Property "IpId"

| | |
|---|---|
| NAME | IpId |
| DESCRIPTION | This property presents the Identification field of the IP packet header |
| SYNTAX | unit16 |

#### 7.1.30.23 Property "IpOption"

| | |
|---|---|
| NAME | IpOption |
| DESCRIPTION | This property presents the Option field of the IP packet header |
| SYNTAX | string |

#### 7.1.30.24 Property "IpFragBit"

| | |
|---|---|
| NAME | IpFragBit |
| DESCRIPTION | This property presents the FragmentBit field of the IP packet header |
| SYNTAX | string |

#### 7.1.30.25 Property "IpFragBitMode"

| | |
|---|---|
| NAME | IpFragBitMode |
| DESCRIPTION | This property presents a flag indicating FragmentBit field of the IP packet header |
| SYNTAX | unit16 |
| VALUES | Normal(0), All(1), Any(2), Not(3) |

#### 7.1.30.26 Property "IpDsize"

| | |
|---|---|
| NAME | IpDsize |
| DESCRIPTION | This property presents the size of the payload field of a packet |
| SYNTAX | unit16 |

#### 7.1.30.27 Property "IpDsizeFlag"

| | |
|---|---|
| NAME | IpDsizeFlag |
| DESCRIPTION | This property presents a flag indicating the property of IpDsize |
| SYNTAX | unit16 |
| VALUES | EQ(0), GT(1), LT(2), RG(3) |

#### 7.1.30.28 Property "TcpFlag"

| | |
|---|---|
| NAME | TcpFlag |
| DESCRIPTION | This property indicates the Flag field of the TCP packet header |
| SYNTAX | string |

#### 7.1.30.29 Property "TcpFlagMode"

| | |
|---|---|
| NAME | TcpFlagMode |
| DESCRIPTION | This property presents a flag indicating the Flag field of the TCP packet header |
| SYNTAX | unit16 |
| VALUES | Normal(0), All(1), Any(2), Not(3) |

#### 7.1.30.30 Property "Flow"

| | |
|---|---|
| NAME | Flow |
| DESCRIPTION | This property is used to define the packet flow |
| SYNTAX | string |

#### 7.1.30.31 Property "Flowbits"

| | |
|---|---|
| NAME | Flowbits |
| DESCRIPTION | This property is used to trace the status of flow through the transmission protocol session |
| SYNTAX | string |

#### 7.1.30.32 Property "TcpSeq"

| | |
|---|---|
| NAME | TcpSeq |
| DESCRIPTION | This property presents the value of SEQ field of the TCP packet header |
| SYNTAX | uint32 |

#### 7.1.30.33 Property "TcpAck"

| | |
|---|---|
| NAME | TcpAck |
| DESCRIPTION | This property presents the value of the ACK field of the TCP packet header |
| SYNTAX | uint32 |

#### 7.1.30.34 Property "Window"

| | |
|---|---|
| NAME | Window |
| DESCRIPTION | This property is used to check the size of the TCP window |
| SYNTAX | uint32 |

#### 7.1.30.35 Property "IcmpType"

| | |
|---|---|
| NAME | IcmpType |
| DESCRIPTION | This property presents the value of ID field of the ICMP packet header |
| SYNTAX | uint16 |

#### 7.1.30.36 Property "IcmpCode"

| | |
|---|---|
| NAME | IcmpCode |
| DESCRIPTION | This property presents the value of Code field of the ICMP packet header |
| SYNTAX | uint16 |

#### 7.1.30.37 Property "IcmpId"

| | |
|---|---|
| NAME | IcmpId |
| DESCRIPTION | This property presents the value of ID field of the ICMP packet header |
| SYNTAX | uint16 |

#### 7.1.30.38 Property "IcmpSeq"

| | |
|---|---|
| NAME | IcmpSeq |
| DESCRIPTION | This property presents the value of Sequence Number field of the ICMP packet header |
| SYNTAX | uint16 |

#### 7.1.30.39 Property "Rpc"

| | |
|---|---|
| NAME | Rpc |
| DESCRIPTION | This property is used to check the RPC application, version, and procedure number from the SUNRPC CALL Request |
| SYNTAX | string |

#### 7.1.30.40 Property "IpProto"

| | |
|---|---|
| NAME | IpProto |
| DESCRIPTION | This property presents the value of Protocol field of the IPv4 packet header or the value of the Next Header field of the IPv6 packet header |
| SYNTAX | unit16 |
| VALUES | ICMP(1), IGMP(2), TCP(6), IGRP(9), UDP(17), GRE(47), ESP(50), AH(51), EIGRP(88), OSPF(89), L2TP(115) |

#### 7.1.30.41 Property "IpProtoFlag"

| | |
|---|---|
| NAME | IpProtoFlag |
| DESCRIPTION | This property represents the flag of the IpProto property |
| SYNTAX | unit16 |
| VALUES | GT(1), LT(2) |

#### 7.1.30.42 Property "IpProtoNegated"

| | |
|---|---|
| NAME | IpProtoNegated |
| DESCRIPTION | This property represents the negation of the IpProto property |
| SYNTAX | boolean |

#### 7.1.30.43 Property "Content"

| | |
|---|---|
| NAME | Content |
| DESCRIPTION | This property is used to search for specific content in the packet payload and trigger response based on such data |
| SYNTAX | string |

#### 7.1.30.44 Property "IpDataLength"

| | |
|---|---|
| NAME | IpDataLength |
| DESCRIPTION | This property represents the data size of the IP packet |
| SYNTAX | uint16 |

#### 7.1.30.45 Property "IpDataLengthFlag"

| | |
|---|---|
| NAME | IpDataLengthFlag |
| DESCRIPTION | This property represents the flag of the IpDataLength property |
| SYNTAX | uint16 |
| VALUES | GT(1), LT(2) |

#### 7.1.30.46 Property "UdpDataLength"

| | |
|---|---|
| NAME | UdpDataLength |
| DESCRIPTION | This property is used to represent the data size of the UDP packet |
| SYNTAX | uint16 |

#### 7.1.30.47 Property "UdpChecksum"

| | |
|---|---|
| NAME | UdpChecksum |
| DESCRIPTION | This property is used to represent the checksum field of the UDP packet |
| SYNTAX | uint16 |

#### 7.1.30.48 Property "IcmpChecksum"

| | |
|---|---|
| NAME | IcmpChecksum |
| DESCRIPTION | This property is used to represent the checksum field of the ICMP packet |
| SYNTAX | uint16 |

#### 7.1.30.49 Property "CountThreshold"

| | |
|---|---|
| NAME | CountThreshold |
| DESCRIPTION | This property represents the count threshold value |
| SYNTAX | uint32 |

#### 7.1.30.50 Property "TimeThreshold"

| | |
|---|---|
| NAME | TimeThreshold |
| DESCRIPTION | This property represents the time threshold value |
| SYNTAX | uint16 |

### 7.1.31 Class "ExtendedHeaderFilter"

| | |
|---|---|
| NAME | ExtendedHeaderFilter |
| DESCRIPTION | This class includes the extended header of the IPv6 packet |
| DERIVED FROM | FilterEntryBase |
| PROPERTIES | HBHLength, HBHLengthFlag, HBHopType, HBHopLength, HBHopLengthFlag, HBHopData, DO1Length, DO1LengthFlag, DO1opType, DO1opLength, DO1opLengthFlag, DO1opData, RHLength, RHLengthFlag, RHType, RHSegLeft, RHAddrs, FHOffset, FHOffsetFlag, FHMFlag, FHID, FHIDFlag, AHPayloadLength, AHSPI, AHSPIFlag, AHSeq, AHSeqFlag, AHAuthData, ESPSPI, ESPSPIFlag, ESPSeq, ESPSeqFlag, ESPAuthData, DO2Length, DO2LengthFlag, DO2opType, DO2opLength, DO2opLengthFlag, DO2opData |

#### 7.1.31.1 Property "HBHLength"

| | |
|---|---|
| NAME | HBHLength |
| DESCRIPTION | This property represents the length of hop-by-hop header |
| SYNTAX | uint8 |

#### 7.1.31.2 Property "HBHLengthFlag"

| | |
|---|---|
| NAME | HBHLengthFlag |
| DESCRIPTION | This property represents the flag of HBHLength property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

#### 7.1.31.3 Property "HBHopType"

| | |
|---|---|
| NAME | HBHopType |
| DESCRIPTION | This property represents the option type field of hop-by-hop header |
| SYNTAX | uint8 |

#### 7.1.31.4 Property "HBHopLength"

| | |
|---|---|
| NAME | HBHopLength |
| DESCRIPTION | This property represents the option data length field of hop-by-hop header |
| SYNTAX | uint8 |

### 7.1.31.5 Property "HBHopLengthFlag"

| | |
|---|---|
| NAME | HBHopLengthFlag |
| DESCRIPTION | This property represents the flag of HBHopLength property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

### 7.1.31.6 Property "HBHopData"

| | |
|---|---|
| NAME | HBHopData |
| DESCRIPTION | This property represents the option data field of hop-by-hop header |
| SYNTAX | string |

### 7.1.31.7 Property "DO1Length"

| | |
|---|---|
| NAME | DO1Length |
| DESCRIPTION | This property represents the length of the Destination Option header |
| SYNTAX | uint8 |

### 7.1.31.8 Property "DO1LengthFlag"

| | |
|---|---|
| NAME | DO1LengthFlag |
| DESCRIPTION | This property represents the flag of the DO1Length property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

### 7.1.31.9 Property "DO1opType"

| | |
|---|---|
| NAME | DO1opType |
| DESCRIPTION | This property represents the Option Type field of the Destination Option header |
| SYNTAX | uint8 |

### 7.1.31.10 Property "DO1opLength"

| | |
|---|---|
| NAME | DO1opLength |
| DESCRIPTION | This property represents the Option Data Length field of the Destination Option header |
| SYNTAX | uint8 |

### 7.1.31.11 Property "DO1opLengthFlag"

| | |
|---|---|
| NAME | DO1opLengthFlag |
| DESCRIPTION | This property represents the flag of the DO1opLength property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

### 7.1.31.12 Property "DO1opData"

| | |
|---|---|
| NAME | DO1opData |
| DESCRIPTION | This property represents the Option Data field of the Destination Option header |
| SYNTAX | uint8 |

### 7.1.31.13 Property "RHLength"

| | |
|---|---|
| NAME | RHLength |
| DESCRIPTION | This property represents the length of the Routing header |
| SYNTAX | uint8 |

### 7.1.31.14 Property "RHLengthFlag"

| | |
|---|---|
| NAME | RHLengthFlag |
| DESCRIPTION | This property represents the flag of the Routing header |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

### 7.1.31.15 Property "RHType"

| | |
|---|---|
| NAME | RHType |
| DESCRIPTION | This property represents the Routing Type field of the Routing header |
| SYNTAX | uint8 |

### 7.1.31.16 Property "RHSegLeft"

| | |
|---|---|
| NAME | RHSeqLeft |
| DESCRIPTION | This property represents the Segments Left field of the Routing header |
| SYNTAX | uint8 |

#### 7.1.31.17 Property "RHAddrs"

| | |
|---|---|
| NAME | RHAddrs |
| DESCRIPTION | This property represents the type-specific data field of the Destination header |
| SYNTAX | uint8 |

#### 7.1.31.18 Property "FHOffset"

| | |
|---|---|
| NAME | FHOffset |
| DESCRIPTION | This property represents the Fragment Offset field of the Fragment header |
| SYNTAX | uint13 |

#### 7.1.31.19 Property "FHOffsetFlag"

| | |
|---|---|
| NAME | FHOffsetFlag |
| DESCRIPTION | This property represents the flag of the FHOffset property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

#### 7.1.31.20 Property "FHMFlag"

| | |
|---|---|
| NAME | FHMFlag |
| DESCRIPTION | This property represents the M flag field of the Fragment header |
| SYNTAX | uint1 |
| VALUES | more fragments(1), last fragment(0) |

#### 7.1.31.21 Property "FHID"

| | |
|---|---|
| NAME | FHID |
| DESCRIPTION | This property represents the Identification field of the Fragment header |
| SYNTAX | uint32 |

#### 7.1.31.22 Property "FHIDFlag"

| | |
|---|---|
| NAME | FHIDFlag |
| DESCRIPTION | This property represents the flag of the FHID property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

#### 7.1.31.23 Property "AHPayloadLength"

| | |
|---|---|
| NAME | AHPayloadLength |
| DESCRIPTION | This property represents the Payload Length field of the Authentication header |
| SYNTAX | uint8 |

#### 7.1.31.24 Property "AHSPI"

| | |
|---|---|
| NAME | AHSPI |
| DESCRIPTION | This property represents the Security Parameters Index field of the Authentication header |
| SYNTAX | uint32 |

#### 7.1.31.25 Property "AHSPIFlag"

| | |
|---|---|
| NAME | AHSPIFlag |
| DESCRIPTION | This property represents the flag of the AHSPI property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

#### 7.1.31.26 Property "AHSeq"

| | |
|---|---|
| NAME | AHSeq |
| DESCRIPTION | This property represents the Sequence Number field of the Authentication header |
| SYNTAX | uint32 |

#### 7.1.31.27 Property "AHSeqFlag"

| | |
|---|---|
| NAME | AHSeqFlag |
| DESCRIPTION | This property represents the flag of the AHSeq property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

#### 7.1.31.28 Property "AHAuthData"

| | |
|---|---|
| NAME | AHAuthData |
| DESCRIPTION | This property represents the Authentication Data of the Authentication header |
| SYNTAX | string |

### 7.1.31.29 Property "ESPSPI"

| | |
|---|---|
| NAME | ESPSPI |
| DESCRIPTION | This property represents the Security Parameters Index field of the Encapsulating Security Payload header |
| SYNTAX | uint32 |

### 7.1.31.30 Property "ESPSPIFlag"

| | |
|---|---|
| NAME | ESPSPIFlag |
| DESCRIPTION | This property represents the flag of the ESPSPI property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

### 7.1.31.31 Property "ESPSeq"

| | |
|---|---|
| NAME | ESPSeq |
| DESCRIPTION | This property represents the Sequence Number field of the Encapsulating Security Payload header |
| SYNTAX | uint32 |

### 7.1.31.32 Property "ESPSeqFlag"

| | |
|---|---|
| NAME | ESPSeqFlag |
| DESCRIPTION | This property represents the flag of the ESPSeq property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

### 7.1.31.33 Property "ESPAuthData"

| | |
|---|---|
| NAME | ESPAuthData |
| DESCRIPTION | This property represents the Authentication Data field of the Encapsulating Security Payload header |
| SYNTAX | string |

### 7.1.31.34 Property "DO2Length"

| | |
|---|---|
| NAME | DO2Length |
| DESCRIPTION | This property represents the length of the Destination Option header |
| SYNTAX | uint8 |

#### 7.1.31.35 Property "DO2LengthFlag"

| | |
|---|---|
| NAME | DO2LengthFlag |
| DESCRIPTION | This property represents the flag of the DO2Length property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

#### 7.1.31.36 Property "DO2opType"

| | |
|---|---|
| NAME | DO2opType |
| DESCRIPTION | This property represents the Option Type field of the Destination Option header |
| SYNTAX | uint8 |

#### 7.1.31.37 Property "DO2opLength"

| | |
|---|---|
| NAME | DO2opLength |
| DESCRIPTION | This property represents the Option Data Length field of the Destination Option header |
| SYNTAX | uint8 |

#### 7.1.31.38 Property "DO2opLengthFlag"

| | |
|---|---|
| NAME | DO2opLengthFlag |
| DESCRIPTION | This property represents the flag of the DO2opLength property |
| SYNTAX | uint8 |
| VALUES | =(1), !=(2), >(3), <(4), >=(5), <=(6) |

#### 7.1.31.39 Property "DO2opData"

| | |
|---|---|
| NAME | DO2opData |
| DESCRIPTION | This property represents the Option Data field of the Destination Option header |
| SYNTAX | string |

## 7.2 The class hierarchy of NSPIM

The inheritance hierarchy for structural classes consisting of NSPIM is shown in Figure 1.

```
ManagedElement (abstract) [c]
 |
 +-- Policy (abstract) [p]
 |    |
 |    +-- PolicySet (abstract) [e]
 |    |    |
 |    |    +-- PolicyGroup [p]
 |    |    |
 |    |    +-- PolicyRule [p]
 |    |         |
 |    |         +-- AttackDetectionRule [n]
 |    |         |
 |    |         +-- AccessControlRule [n]
 |    |         |
 |    |         +-- TrafficControlRule [n]
 |    |         |
 |    |         +-- RoutingControlRule [n]
 |    |         |
 |    |         +-- AlertControlRule [n]
 |    |
 |    +-- PolicyCondition (abstract) [p]
 |    |    |
 |    |    +-- PolicyTimePeriodCondition [p]
 |    |    |
 |    |    +-- SimplePolicyCondition [e]
 |    |    |
 |    |    +-- CompoundPolicyCondition [e]
 |    |    |    |
 |    |    |    +-- AccessControlCondition [n]
 |    |    |    |
 |    |    |    +-- TrafficControlCondition [n]
 |    |    |    |
 |    |    |    +-- RoutingControlCondition [n]
 |    |    |    |
 |    |    |    +-- AlertControlCondition [n]
 |    |    |
 |    |    +-- PacketMonitoringCondition [n]
 |    |
 |    +-- PolicyAction (abstract) [p]
 |    |    |
 |    |    +-- ResponseAction [n]
 |    |    |
 |    |    +-- AccessControlAction [n]
 |    |    |
 |    |    +-- TrafficControlAction [n]
 |    |    |
 |    |    +-- BlackHoleAction [n]
 |    |    |
 |    |    +-- AlertControlAction [n]
 |    |         |
 |    |         +-- AlertFilteringAction [n]
 |    |         |
 |    |         +-- AlertSamplingAction [n]
 |    |
 |   +-- PolicyVariable (abstract) [e]
 |   |   |
 |   |   +-- PolicyExplicitVariable [e]
 |   |   |   |
 |   |   |   +-- PolicyAttackIDVariable [n]
 |   |   |
 |   |   +-- PolicyImplicitVariable [e]
 |   |       |
 |   |       +-- PolicySrcIPv4Variable [e]
 |   |       |
 |   |       +-- PolicyDstIPv4Variable [e]
 |   |       |
 |   |       +-- PolicySrcPortVariable [e]
 |   |       |
 |   |       +-- PolicyDstPortVariable [e]
 |   |       |
 |   |       +-- PolicyProtocolVariable [e]
 |   |       |
 |   |       +-- PolicyFlowDirectionVariable [e]
 |   |       |
 |   |       +-- PolicyTCPUrgVariable [n]
 |   |       |
 |   |       +-- PolicyTCPAckVariable [n]
 |   |       |
 |   |       +-- PolicyTCPPshVariable [n]
 |   |       |
 |   |       +-- PolicyTCPRstVariable [n]
 |   |       |
 |   |       +-- PolicyTCPSynVariable [n]
 |   |       |
 |   |       +-- PolicyTCPFinVariable [n]
 |   |       |
 |   |       +-- PolicyTCPRes1Variable [n]
 |   |       |
 |   |       +-- PolicyTCPRes2Variable [n]
 |   |       |
 |   |       +-- PolicyICMPTypeVariable [n]
 |   |       |
 |   |       +-- PolicyICMPCodeVariable [n]
 |   |
 |   +-- PolicyValue (abstract) [e]
 |       |
 |       +-- PolicyIPv4AddrValue [e]
 |       |
 |       +-- PolicyStringValue [e]
 |       |
 |       +-- PolicyIntegerValue [e]
 |       |
 |       +-- PolicyBooleanValue [e]
 |
 +- Collection (abstract) [c]
 |
 +- ManagedSystemElement (abstract) [c]
    |
    +- LogicalElement (abstract) [c]
       |
       +- System (abstract) [c]
       |   |
       |  +- AdminDomain (abstract) [c]
       |       |
       |      +- ReusablePolicyContainer[e]
       |       |
       |      +- ComputerSystem [c]
       |          |
       |         +- UnitaryComputerSystem [c]
       |
       +- FilterEntryBase (abstract) [c]
       |   |
       |  +- DetailedPacketFilter [n]
       |   |
       |  +- ExtendedHeaderFilter [n]
       |
       +- FilterList [c]

     [c] CIM
     [p] PCIM
     [e] PCIMe
     [n] NSPIM
```

**Figure 1 – NSPIM class inheritance hierarchy**

NSPIM includes five sub-rules, five sub-conditions, and six sub-actions to define the network security policies that cannot be expressed by classes of CIM, PCIM and PCIMe.

## 7.3 Association and aggregation of NSPIM

### 7.3.1 Class definitions

#### 7.3.1.1 Association "FilterOfMonitoringCondition"

| | |
|---|---|
| NAME | FilterOfMonitoringCondition |
| DESCRIPTION | This association class represents the association of FilterList and PacketMonitoringCondition. |
| DERIVED FROM | Dependency |

### 7.3.2 The hierarchy of NSPIM associated classes

In NSPIM, instances of structural classes are associated with association classes. Specifically, a rule consists of a set of condition and a set of action. A condition and an action consist of one or more variables and corresponding values. The policy association classes defined in NSPIM are sufficient and necessary to allow them to represent policies related to each other. The hierarchy of several association classes of NSPIM is shown in Figure 2.

```
[unrooted]
    |
   +---PolicyComponent (abstract)
    |    |
    |   +---PolicySetComponent [e]
    |    |
    |   +---PolicyConditionStructure (abstract) [e]
    |    |    |
    |    |   +---PolicyConditionInPolicyRule [p]
    |    |    |
    |   |   +---PolicyConditionInPolicyCondition [e]
    |   |
    |   +---PolicyRuleValidityPeriod [p]
    |   |
    |   +---PolicyActionStructure (abstract) [e]
    |   |   |
    |   |   +---PolicyActionInPolicyRule [p]
    |   |   |
    |   |   +---PolicyActionInPolicyAction [e]
    |   |
    |   +---PolicyVariableInSimplePolicyCondition [e]
    |   |
    |   +---PolicyValueInSimplePolicyCondition [e]
    |   |
    |   +---PolicyVariableInSimplePolicyAction [e]
    |   |
    |  +---PolicyValueInSimplePolicyAction [e]
    |
   +---Dependency (abstract) [c]
    |   |
    |   +---FilterOfMonitoringCondition [n]
    |   |
    |   +---PolicyInSystem (abstract) [p]
    |   |   |
    |   |   +---PolicySetInSystem (abstract) [e]
    |   |   |   |
    |   |   |   +---PolicyGroupInSystem [p]
    |   |   |   |
    |   |   |   +---PolicyRuleInSystem [p]
    |   |   |
    |   |   +---ReusablePolicy [e]
    |   |
    |   +---ExpectedPolicyValuesForVariable [e]
    |   |
    |   +---PolicyRoleCollectionInSystem [e]
    |
   +---Component (abstract) [c]
    |   |
    |   +---SystemComponent [c]
    |   |   |
    |   |   +---ContainedDomain [e]
    |   |
    |   +---EntriesInFilterList [e]
    |
   +---MemberOfCollection [e]
       |
       +--- ElementInPolicyRoleCollection [e]
```

[c] CIM
[p] PCIM
[e] PCIMe
[n] NSPIM

**Figure 2 – NSPIM association class inheritance hierarchy**

## 7.4 Relation of Classes

**Table 2 – Relation symbols**

| Relation | Meaning |
|---|---|
| ↑ | Inheritance |
| — | Association |
| —◇ | Aggregation |
| * | Equivalent to: 0..n |

X.1036(07)_T02

### 7.4.1 PolicyRule

A PolicyCondition class as well as PolicyAction are needed to make PolicyRule. A PolicyCondition class is associated with the PolicyRule class via the aggregation of PolicyConditionInPolicyCondition, and a PolicyAction class, with the PolicyRule class via the aggregation of PolicyActionInPolicyAction.

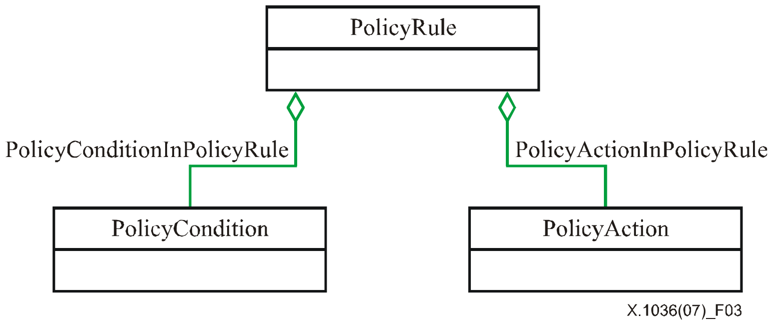

**Figure 3 – Conceptual model of PolicyRule**

### 7.4.2 PolicyCondition

PolicyCondition consists of many SimplePolicyCondition classes associated with the PolicyVariable class via the aggregation of the PolicyVariableInSimplePolicyCondition class. Each PolicyVariable class is coupled with a PolicyValue class via the association of ExpectedPolicyValuesForVariable.

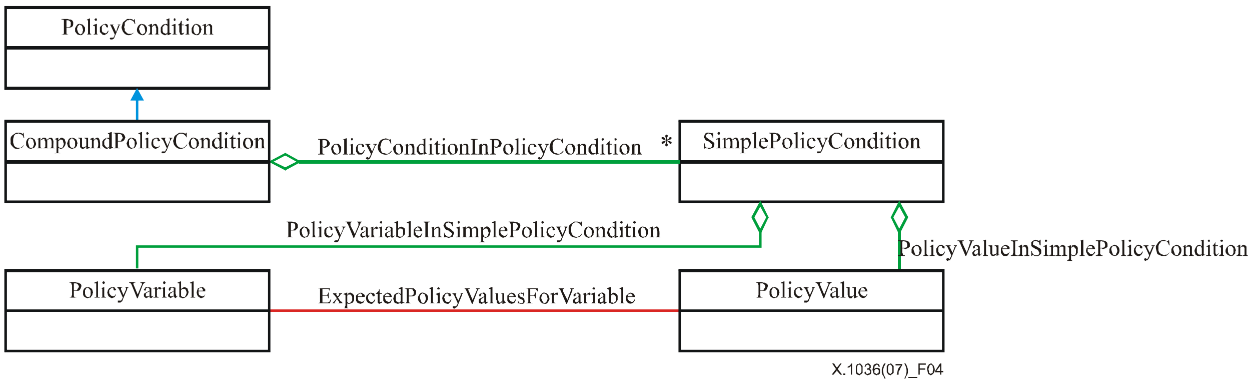
PolicyCondition
CompoundPolicyCondition
PolicyConditionInPolicyCondition
*
SimplePolicyCondition
PolicyVariableInSimplePolicyCondition
PolicyValueInSimplePolicyCondition
PolicyVariable
ExpectedPolicyValuesForVariable
PolicyValue
X.1036(07)_F04

**Figure 4 – Conceptual model of the PolicyCondition**

### 7.4.3 FilterList

The alternative method for making a policy condition is to use a FilterList class. Expressing packet filters such that they are mapped more directly to the packet fields to which the filters are being applied is useful. A FilterList class aggregates instances of FilterEntryBase via the aggregation of EntriesInFilterList.

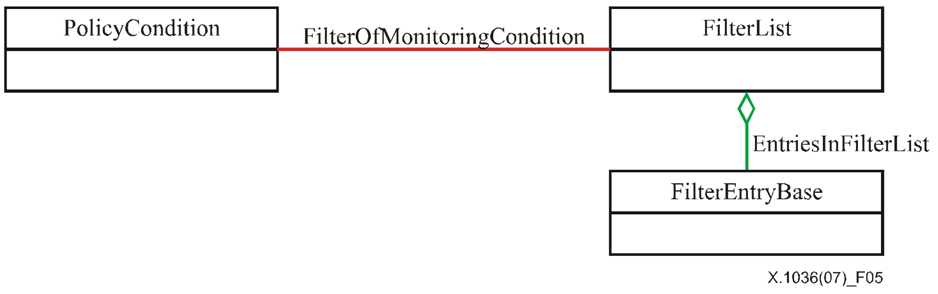

**Figure 5 – Conceptual model of a FilterList**

## 8 Network security policy framework

NSPIM is defined as a network security policy framework for the representation, creation, storage, and distribution of policies on network security and response in a network security system. NSPIM forces each component of the network security policy framework to be flexible and extensible. The framework includes the operational structure of policy management unit (PMU) and data schema of policy repository (PR). In addition, policy provision objects between policy decision unit (PDU) and policy enforcement system (PES) can be defined based on NSPIM. The components of the network security policy framework (based on NSPIM) are shown in Figure 6.

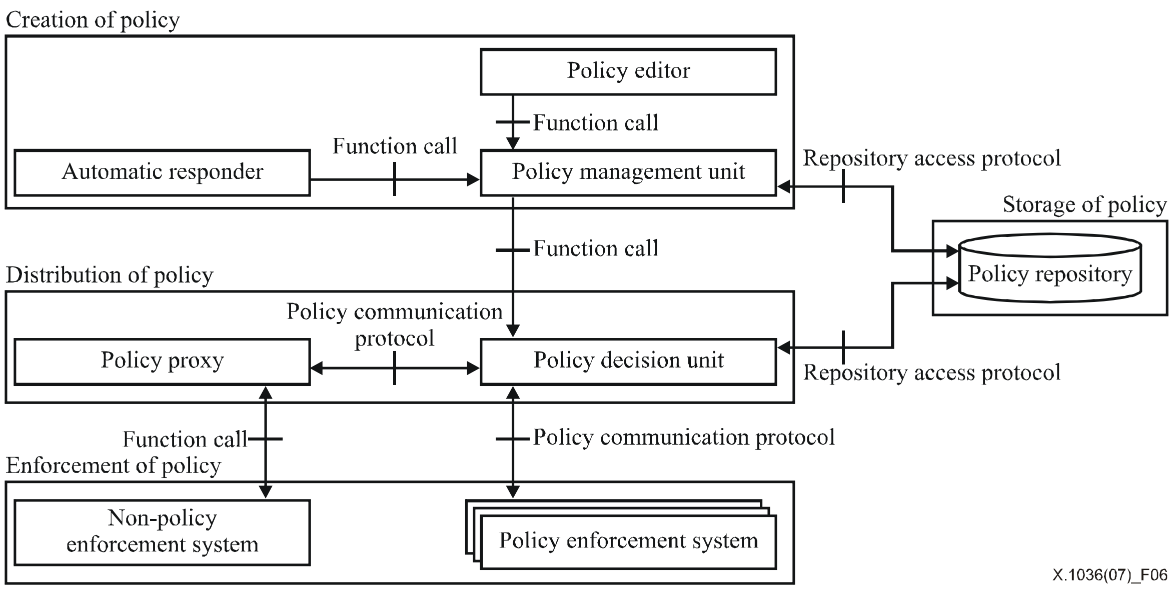

**Figure 6 – Network security policy framework**

## 8.1 Creation of policies

### 8.1.1 Components

#### 8.1.1.1 Policy editor

The policy editor serves as the interface between the system and the network administrator to construct policies, deploy policies and monitor the status of the policy. The policy editor supports policy editing, policy presentation, rule translation and rule validation functions. In this component, many rules are translated from abstract or human understandable forms into the syntax of the network security policy information model of the repository.

#### 8.1.1.2 Automatic responder

The automatic responder is useful in responding to the abnormal network status immediately. It generates and decides response actions in accordance with security incidents and forwards them to the policy management unit when an incident violating a normal network status is detected.

#### 8.1.1.3 Policy management unit

The policy management unit performs policy validation and checks for conflict with existing policies. It translates the policy rules with the schema of the policy repository and stores policies in the policy repository. The policy management unit also deletes expired policy rules. Much of the functionality of the policy management unit is driven by the policy editor. For instance, the policy management unit translates the policies created at the policy editor and handles communications with the policy repository on behalf of the policy editor. The policy management unit also has the responsibility of notifying the policy decision unit of the changes in policies.

### 8.1.2 Functions

The following basic functions should be performed by the policy editor and policy management unit within a network security policy framework.

#### 8.1.2.1 Policy Editor

a) The policy editor lets the network manager create and edit policies and review past policies. It also works with the policy management unit to translate the policy rules created or edited into a form that is compatible with the schema and storage requirements of the policy repository by a network manager.

b) The policy editor should not only guide a network manager through the creation of new rules and policies through the possible provision of templates; it should also provide immediate feedback when the network manager uses an improper syntax or enters incorrect data types.

#### 8.1.2.2 Policy Management Unit

a) The policy editor drives much of the functionality of the policy management unit. For instance, the policy management unit translates the policies created at the policy editor and handles communications with the policy repository on behalf of the policy editor. In some cases, the network manager will use the policy editor to retrieve existing policies or policy rules for editing. In this case, the policy management unit will have to retrieve the required policies from the repository.

b) Whereas the policy editor will most likely perform syntax and data-typing checks, the policy management unit works with the policy editor to check the policies for conflicts.

c) The management tool also has the responsibility of notifying the policy decision unit (PDU) of the changes in policies.

### 8.1.3 Requirements

The following are the general requirements for the policy editor and policy management unit:

a) *User-friendly interface*

A user-friendly interface to a policy-based networking system should not only include graphical representations of policy hierarchies, but also a form fill-out mode as well as templates for creating or editing policies.

b) *Conflict checking capabilities*

The policy editor and policy management unit not only need to check the syntax and data typing but must also alert the network manager of policy conflicts when policies are created.

c) *Identifying policy*

Since a network manager cannot always be certain what the effects of the new policies will be on the network, the policy editor must provide some way of either naming or at least identifying previous sets of policies.

d) *Security*

The policy editor also acts as the first level of security for the network by authenticating each user (i.e., network manager) of the system as described in [ITU-T X.800].

## 8.2 Storage of policies

### 8.2.1 Components

#### 8.2.1.1 Policy repository

The policy repository is a directory and/or other storage service (e.g., relational database) where policy and related information are stored. The primary purpose of the policy repository is to store the policy rules created for the policy management unit. If a directory is the policy repository, then it can also be used to store other policy-related information such as user IDs, access control lists, and pointers to device or network information, On the other hand, if a database is used to store policy, the system must include some link between the database and the data stores where user information is located and which is usually a directory. Whether the repository is a directory or a database, the structure of the data stored in the repository is determined by the schema developed for the system.

### 8.2.2 Functions

The following are the basic functions required by a policy repository:

a) The policy repository stores policies such that whoever or whatever needs to use the policies can retrieve them. A network manager may use a policy management unit, or a policy decision unit may request for policies on behalf of the network devices it controls.

b) The policy repository has to provide certain services to do the job correctly. For instance, a data repository must maintain the integrity of the data it receives and stores; this means that any transaction that changes data must be complete and verified. Otherwise, the affected data is restored or rolled back to its previous state.

c) The policy repository should include mechanisms to guard its data against unauthorized access and alterations.

### 8.2.3 Requirements

The general requirements for the policy repository should include:

a) *Standard protocol for accessing data*

   Whether the policy repository is a directory or a database, it should support what has become the standard protocol for accessing data.

b) *Distributed data store*

   When using a distributed data store for policies, the data store should include methods for replicating data among the different locations and ensure that data replication does not affect the distribution of policies within the system.

c) *Support for multi-user management including data versioning*

   The repository should support the versioning of data and rollback features to enable changes to be traced back to administrators or policies to be reset to previous versions in case of an error.

d) *Authentication and access control*

   The policy repository must be secure. The policy repository should offer some means of authenticating users as described in [ITU-T X.805], [ITU-T X.810] and [ITU-T X.812], to ensure that only authorized personnel are allowed to access the contents of the repository.

## 8.3 Distribution of policies

### 8.3.1 Components

#### 8.3.1.1 Policy decision unit

The policy decision unit obtains the policy from the policy repository when the policy management unit sends an alert or the policy enforcement system requests for policies. It converts the policy into PIB that can be understood by the policy enforcement system and distributes PIB to the policy enforcement system. The policy decision unit can use the role-based policy decision and distribution mechanism. This mechanism promotes convenience because it lets the administrator manage and transmit policies according to the roles of devices without configuring each one of the network devices manually.

#### 8.3.1.2 Policy proxy

The policy proxy should intervene between the policy decision unit and most network devices. It translates the PIB generated by the policy decision unit into a configuration file that can be understood by the non-policy enforcement system.

### 8.3.2 Functions

The basic functions focus not only on the policy decision point itself but also on the way PDU interacts with a policy enforcement system.

#### 8.3.2.1 Policy decision unit

a) The policy decision unit must do most of the difficult work of translating policies into information that makes sense to the network devices that should enforce the policies.

b) The policy decision unit is an intermediary in translating higher-level abstractions of policies into configurations that are of use to the policy enforcement system.

c) It can serve as the central point for distributing policies to the policy enforcement system especially in three-tiered architectures.

d) It makes decisions based on policy rules and state of services managed by those policies. It is also responsible for policy rule interpretation and initiating deployment.

#### 8.3.2.2 Policy proxy

In general, network devices are not very intelligent especially when it comes to dealing with changing network conditions. Therefore, some additional translator module must intervene between the PDU and most of today's network devices – this is the role of the policy proxy.

a) The policy proxy is responsible for converting the policies generated by PDU into configuration files that can be understood by the non-policy enforcement system.

b) It will translate policies into network communication commands (e.g., CLI or SNMP) and merely send the configuration data to the non-policy enforcement system.

### 8.3.3 Requirements

The following are the general requirements for the policy decision unit (and any associated policy proxy):

a) *Support for multiple configuration protocols*

   Since a single protocol can hardly take care of all the configuration needs of a network, PDU should be multifaceted when it comes to protocols.

b) *PDU redundancy*

   A policy-based network security system should include backups for PDU; otherwise, there is little assurance that the system will be reliable.

c) *Review of policy translation procedures*

   Whether to assure a curious network manager that the translations are correct or to help someone troubleshoot a device configuration, the system should provide the network editor with a way of reviewing the translations stored in PDU or policy proxy.

d) *Alerts associated with policy conflicts*

   PDU should send alerts to PMU in case of policy conflicts or when a translation cannot be performed.

e) *Awareness of the capabilities of PES*

   PDU should be aware of the operational state as well as the capabilities of the PES it will configure to reach an appropriate decision for configuring PES.

f) *Security*

   PDU must enforce security for communications between itself and the PES it controls as well as with the policy repository. PDU and PES must be able to authenticate themselves with each other using at least a shared key and a message digest. Messages transmitted between the two components should also be encrypted.

## 8.4 Enforcement of policies

### 8.4.1 Components

#### 8.4.1.1 Policy enforcement system

The policy enforcement system can understand and enforce the policy rules delivered from the policy decision unit. It is a network device (gateway, switch, end-host) that requests for and applies policy-based decisions from one or more PDUs. Policy enforcement point (PEP) is an agent running on or within a resource (e.g., device) that enforces a policy decision and/or makes a configuration change.

#### 8.4.1.2 Non-policy enforcement system

Network devices can be categorized into groups: policy-aware and policy-unaware. For policy-aware devices, PES is a component residing in those devices. Note, however, that policy-unaware devices are those that are unable to interpret any portion of the policy information. Most of the networking devices cannot understand policies and support protocols such as the proposed COPS or LDAP for disseminating policies. Nonetheless, the non-policy enforcement system should be controlled to protect the network. Therefore, the non-policy enforcement system needs to be managed by the network security policy framework.

### 8.4.2 Functions

The following are the basic functions required by the policy enforcement systems (and non-policy enforcement systems):

a) It is responsible for collecting the necessary information on the current network state, traffic situation, and transmission errors as well as any relevant information and reporting them to PDU.

b) The policy enforcement system utilizes the commands it receives from PDU (directly or via a policy proxy) to process network traffic.

### 8.4.3 Requirements

The general requirements for policy enforcement systems (and non-policy enforcement systems) should include the following:

a) *Reliability*

The sensitivity of the network security policy framework necessitates reliable operation. A typical requirement for the policy enforcement systems within both LAN and WAN is to provide end-to-end connectivity to all devices within the network.

b) *Small delays*

The timing requirements of policy decisions related to policy communication protocols are expected to be quite strict. The PES-to-PDU protocol should add a small amount of delay to the response delay experienced by queries placed by PES to PDU.

c) *Support for PES-initiated, two-way transactions*

The protocol must allow for two-way transactions (request-response exchanges) between PES and PDU. In particular, PESs must be able to initiate requests for policy decision, re-negotiation of previously made policy decision, and exchange of policy information.

d) *Support for asynchronous notification*

PDU sends asynchronous notifications to PES if necessary to change earlier decisions, generate errors, etc.

e) *Support for many styles of policies*

The designed mechanisms must include support for many policies and policy configurations including viable policies based on the notion of relative priority. These network devices may also need to support different services according to their location in the network.

## 8.5 An example of policy conversion and distribution

We define a set of high-level security policies that describe the behaviour of the network and then describe how these policies are translated from one level of abstraction to another according to the different components of a security policy-based networking system until we finally get to the device configurations for PESs on the network.

A network manager would use the policy editor to write and edit policies and monitor the status of the policy. The policy management unit works in conjunction with the policy editor to translate the policy rules created by managers in the editor into entries matching a predefined schema for storage within the policy repository. The policy repository stores the rules and policies required by the system. As the next component in Figure 7, the policy decision unit (PDU) is responsible for accessing the policy data stored in the repository and making decisions based on those policies. PDU receives requests from network devices or applications, decides on those requests, stores policies in the policy repository, and changes the network conditions. PDU may need to include a translator module called policy proxy to convert policy decisions into commands understandable to older devices that are not policy-aware. As the remaining components in Figure 7, policy enforcement systems (PESs) are network devices that actually implement the decisions passed to them by the policy decision units.

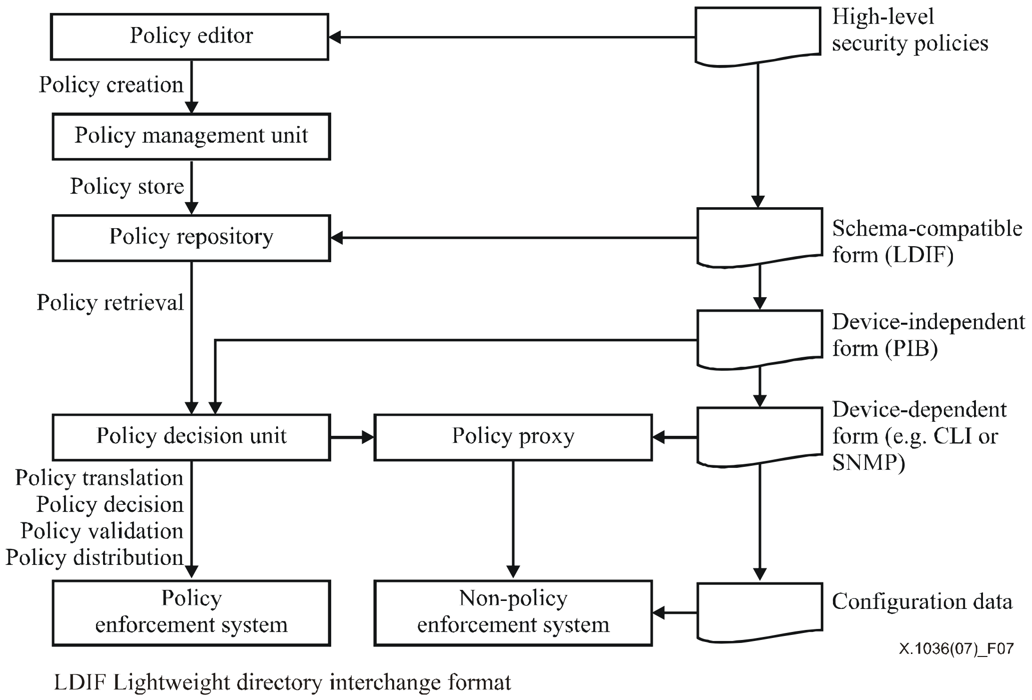

**Figure 7 – Policy conversion and distribution**

Figure 7 illustrates the main steps in distributing and translating policies from the policy editor down to PESs. In basic terms, policies consist of conditions and corresponding actions. The basic building block of a policy is a policy rule (or simply rule), a simple declarative statement associating a policy object with a value. Policy rules define either conditions or actions. Each policy includes one or more conditions and one or more actions.

### 8.5.1 Conversion to policy schema representation

Once the network manager defines the appropriate policies in the policy editor, the policy management unit translates them into a form that is compatible with the schema defined for the policy repository (see Figure 7). One advantage of this approach is the ability to reuse objects such as policy conditions.

Converting the original policies entered in the console to forms that are compatible with the repository's schema requires translating some of the abstractions within the policies. The policies should be grouped together to form a policy group. Once they are grouped, the policy management unit can examine the policies to determine whether there are conflicts between policies.

The policies also need to be associated with PESs, e.g., router. Once this association is created, each policy may be tested for any conflict with other policies. A policy can now be deployed to one or more policy decision units. The policy management unit would send a notification to the PDUs, i.e., there is a new policy for them. This notification message may include references to the affected PESs as well as information on where to find the policy in the repository.

### 8.5.2 Distribution of policy-based configurations

PDUs retrieve the policies from the repository and notify the policy management unit of receipt. Each PDU may also verify the validity of each new policy at this point.

PDUs would perform the appropriate actions for each PES to instantiate the policy on each PES associated with the policy and for which such PDU is responsible. PDU would then provide status information to the policy management unit regarding the success of the deployment operation.

To allow an existing device without a concept of the policy schema to use a policy, PDU would use a policy proxy to provide the appropriate mapping from policy data to device configuration actions. This may involve operations that do not directly map to device capabilities such as handling time and date-related conditions that are not supported by many PESs today.

Prior to deploying the policy-based configuration to the device, i.e., PES, PDU would determine the current configuration. In case of a configuration containing a feature that runs counter to the operation of such PES, PDU would provide feedback to the policy management unit regarding the condition and would refrain from deploying the policy. Otherwise, PDU would issue commands (e.g., SNMP set commands, telnet/CLI, etc., as appropriate for the device) to delete the current configuration or free resources that will no longer be used. At this point, PDU will actually send the configuration commands to the device to enable PES to act in accordance with the policy.

Once the policy-based configuration has been sent to PES, PDU would determine the success of the deployment and provide feedback to the policy management unit. To determine the success in this example, PDU would query the device and examine the information related to the configuration of the PESs to determine whether the configuration now matches what PDU expects based on the policy data. If no errors were encountered during the deployment, and the configuration is correct, PDU reports success.

## Bibliography

| | |
|---|---|
| [b-IETF RFC 2748] | IETF RFC 2748 (2000), *The COPS (Common Open Policy Service) Protocol*, <http://www.ietf.org/rfc/rfc2748.txt> |
| [b-IETF RFC 3318] | IETF RFC 3318 (2003), *Framework Policy Information Base*, <http://www.ietf.org/rfc/rfc3318.txt> |
| [b-DMTF] | Distributed Management Task Force, Inc (1999), *Common Information Model (CIM) Specification, version 2.2*, <http://www.dmtf.org/standards/cim/cim_spec_v22> |

## SERIES OF ITU-T RECOMMENDATIONS

Printed in Switzerland
Geneva, 2008