Internet Engineering Task Force
INTERNET DRAFT
11-July-2000

Jamie Jason
Intel Corporation

IPsec Configuration Policy Model
draft-ietf-ipsp-config-policy-model-01.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html.

Abstract

This document presents an object-oriented model of IPsec policy designed to:

- facilitate agreement about the content and semantics of IPsec policy
- enable derivations of task-specific representations of IPsec policy such as storage schema, distribution representations, and policy specification languages used to configure IPsec-enabled endpoints

The schema described in this document models the IKE phase one parameters as described in [IKE] and the IKE phase two parameters for the IPsec Domain of Interpretation as described in [COMP, ESP, AH, DOI]. It is based upon the core policy classes as defined in the Policy Core Information Model (PCIM) [PCIM].

Table of Contents

1. Introduction

Internet Protocol security (IPsec) policy may assume a variety of forms as it travels from storage to distribution point to decision point. At each step, it needs to be represented in a way that is convenient for the current task. For example, the policy could exist as, but is not limited to:

- a Lightweight Directory Access Protocol (LDAP) [LDAP] schema in a directory
- an on-the-wire representation over a transport protocol like the Common Object Policy Service (COPS) [COPS, COPSPR]
- a text-based policy specification language [SPSL] suitable for editing by an administrator
- an Extensible Markup Language (XML) document

Each of these task-specific representations should be derived from a canonical representation that precisely specifies the content and semantics of the IPsec policy. The purpose of this document is to abstract IPsec policy into a task-independent representation that is not constrained by any particular task-dependent representation.

This document is organized as follows:

- Section 2 provides a quick introduction to the Unified Modeling Language (UML) graphical notation conventions used in this document.

- Section 3 provides the inheritance hierarchy which describes where the IPsec policy classes fit into the policy class hierarchy already defined by PCIM.

- The remainder of the document describes the classes which make up the IPsec policy model.

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 [KEYWORDS].

2. UML Conventions

For this document, a UML static class diagram was chosen as the canonical representation for the IPsec policy model. The reason behind this decision is that UML provides a graphical, task-independent way to model systems. A treatise on the graphical notation used in UML is beyond the scope of this paper. However, given the use of ASCII drawing for UML static class diagrams, a description of the notational conventions used in this document is in order:

- Boxes represent classes, with class names in brackets ([]) representing a virtual class.

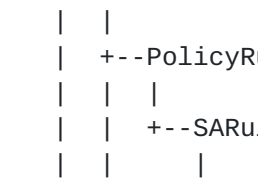
- o A line that terminates with an arrow (<, >, ^, v) denotes inheritance. The arrow always points to the parent class. Inheritance can also be called generalization or specialization (depending upon the reference point). A base class is a generalization of a derived class, and a derived class is a specialization of a base class.
- o Associations are used model a relationship between two classes. Classes that share an association are connected using a line. There are two special kinds of associations - aggregations and compositions. Both model a whole-part relationship between two classes. Associations, and therefore aggregations and compositions, can also be modeled as classes.
- o A line that begins with a "o" denotes aggregation. Aggregation denotes containment in which the contained class and the containing class have independent lifetimes.
- o A line that begins with an "x" denotes composition. Composition denotes containment in which the contained class and the contianing class have coincident lifetimes.
- o Next to a line representing an association appears a multiplicity. Multiplicities indicate the number of objects in the relationship. The multiplicity may be:
  - a range in the form "lower bound..upper bound" indicating the minimum and maximum number of objects.
  - a number that indicates the exact number of objects.
  - an asterisk indicating any number of objects, including zero. Using an asterisk is shorthand for 0..n.
  - the letter n indicating from 1 to many. Using the letter n is shorthand for 1..n.

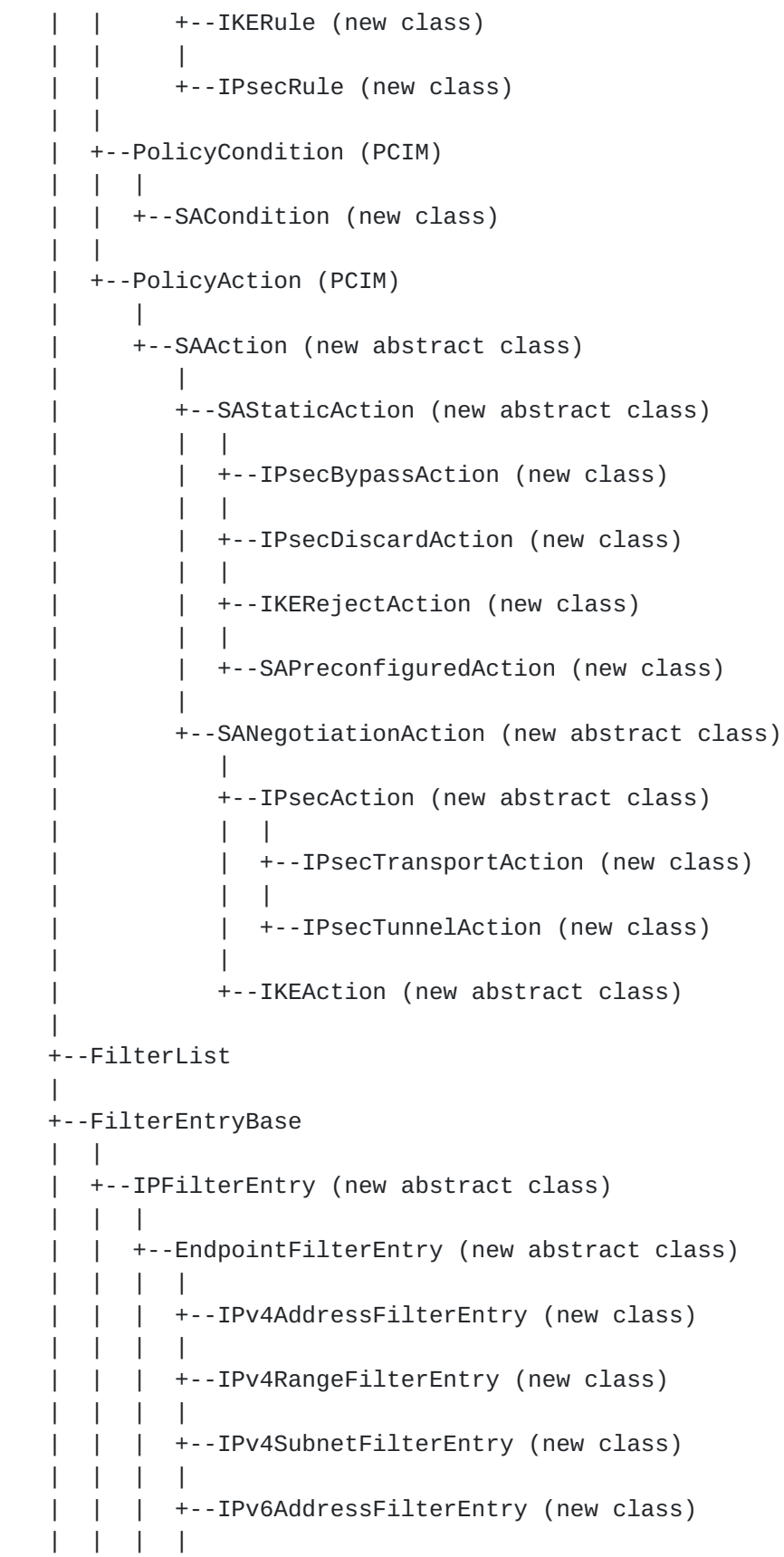
It should be noted that the UML static class diagram presented is a conceptual view of IPsec policy designed to aid in understanding. It does not necessarily get translated class for class into another representation. For example, an LDAP implementation may flatten out the representation to fewer classes (because of the inefficiency of following references).

3. IPsec Policy Model Inheritance Heirarchy

The following diagram represents the inheritance hierarchy and how the IPsec policy model classes fit into PCIM.

```
[unrooted]
|
+--Policy (PCIM)
|  |
|  +--PolicyGroup (PCIM)
|  |  |
|  |  +--IPsecPolicyGroup (new class)
```


| |
| +--PolicyRule (PCIM)
| | |
| | +--SARule (new abstract class)
| | |


+--IKERule (new class)
+--IPsecRule (new class)
+--PolicyCondition (PCIM)
+--SACondition (new class)
+--PolicyAction (PCIM)
+--SAAction (new abstract class)
+--SAStaticAction (new abstract class)
+--IPsecBypassAction (new class)
+--IPsecDiscardAction (new class)
+--IKERejectAction (new class)
+--SAPreconfiguredAction (new class)
+--SANegotiationAction (new abstract class)
+--IPsecAction (new abstract class)
+--IPsecTransportAction (new class)
+--IPsecTunnelAction (new class)
+--IKEAction (new abstract class)
+--FilterList
+--FilterEntryBase
+--IPFilterEntry (new abstract class)
+--EndpointFilterEntry (new abstract class)
+--IPv4AddressFilterEntry (new class)
+--IPv4RangeFilterEntry (new class)
+--IPv4SubnetFilterEntry (new class)
+--IPv6AddressFilterEntry (new class)

```
   |  |  |  +--IPv6RangeFilterEntry (new class)
   |  |  |  |
   |  |  |  +--IPv6SubnetFilterEntry (new class)
   |  |  |  |
   |  |  |  +--FQDNFilterEntry (new class)
```

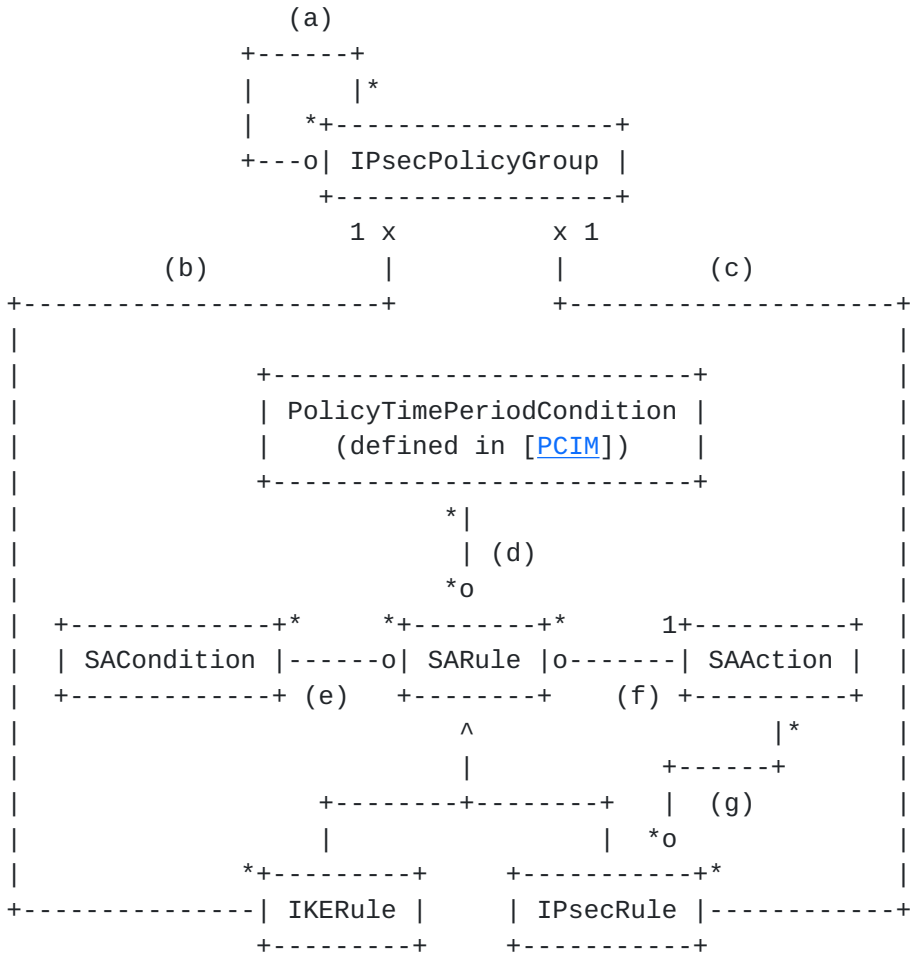
```
|  |  |
|  |  +--PortFilterEntry (new class)
|  |  |
|  |  +--ProtocolFilterEntry (new class)
|  |
|  +--IPSOFilterEntry (new class)
|  |
|  +--CredentialFilterEntry (new class)
|
+--SAProposal (new abstract class)
|  |
|  +--IKEProposal (new class)
|  |
|  +--IPsecProposal (new class)
|
+--SATransform (new abstract class)
   |
   +--AHTransform (new class)
   |
   +--ESPTransform (new class)
   |
   +--IPCOMPTransform (new class)
```

The following diagram represents the inheritance hierarchy and how the IPsec policy model association classes fit into PCIM.

```
[unrooted]
|
+--PolicyGroupInPolicyGroup (PCIM)
|  |
|  +--IPsecPolicyGroupInPolicyGroup (new class)
|
+--PolicyConditionInPolicyRule (PCIM)
|  |
|  +--SAConditionInRule (new class)
|
+--FallbackSAActionInRule (new class)
|
+--EntriesInFilterList (new class)
|
+--ContainedProposal (new class)
|
+--IPsecContainedTransform (new class)
```

4. Policy Classes

The IPsec policy classes represent the set of policies that are contained on a system.



(a) IPsecPolicyGroupInPolicyGroup
(b) RuleForIKENegotiation
(c) RuleForIPsecNegotiation
(d) PolicyRuleValidityPeriod (defined in [PCIM])
(e) SAConditionInRule
(f) SAActionInRule
(g) FallbackSAActionInRule

4.1. The Class IPsecPolicyGroup

The class IPsecPolicyGroup serves as a container of either other IPsecPolicyGroups or a set of IKERules and a set of IPsecRules. Rules contained within an IPsecPolicyGroup MUST have a unique Priority value. The class definition for IPsecPolicyGroup is as

follows:

NAME IPsecPolicyGroup
DESCRIPTION Either a set of IPsecPolicyGroups or a set of IKERules and a set of IPsecRules.

```
DERIVED FROM PolicyGroup (see [PCIM])
ABSTRACT     FALSE
PROPERTIES   PolicyGroupName (from PolicyGroup)
             IKERuleOverridePoint
             IPsecRuleOverridePoint
```

NOTE: for derivations of the schema that are used for policy distribution to an IPsec device (for example, COPS-PR), the server may follow all of IPsecPolicyGroupInPolicyGroup associations and create one policy group which is simply a set of all of the IKE rules and a set of all of the IPsec rules. See the section on the IPsecPolicyGroupInPolicyGroup aggregation for information on merging multiple IPsecPolicyGroups.

4.1.1. The Property IKERuleOverridePoint

This property specifies the rule priority at which the policy author is willing to allow IKERule insertions by a local administrator. For example, the IT department may define the policy on a company-wide basis, but allow groups or individuals to insert rules into the policy to override defaults. Rules are ordered in decreasing order of their priority (i.e., higher priorities come first). The override point specifies that if rules are inserted, they are to be inserted before all rules equal to or less than the override priority value.

For example, assume that there is a group G1 with IKE rules as follows:

```
G1 = { Rule A (priority 50),
       Rule B (priority 25),
       Rule C (priority 15) }
```

The IKE override value for G1 is 20. Now assume that a local administrator wants to insert a set of IKE rules {Rule D, Rule E} where Rule D has a higher priority than Rule E. The new rules will be added before rules in G1 with priority equal to or less than 20. So, when evaluating rules, the order of evaluation would be A, B, D, E, C. Note that the priority of the rules in override set are relative only to the set.

The property is defined as follows:

```
NAME         IKERuleOverridePoint
DESCRIPTION  Specifies the rule priority at which the policy author
             is willing to allow IKERule insertions by a local
             administrator.
SYNTAX       unsigned 16-bit integer
```

4.1.2. The Property IPsecRuleOverridePoint

This property specifies the rule priority at which the policy author is willing to allow IPsecRule insertions by a local administrator.

This property is the same as IKERuleOverridePoint except it is used for the IPsec rules in the IPsecPolicyGroup. The property is defined as follows:

```
NAME         IPsecRuleOverridePoint
DESCRIPTION  Specifies the rule priority at which the policy author
             is willing to allow IPsecRule insertions by a local
             administrator.
SYNTAX       unsigned 16-bit integer
```

4.2. The Class SARule

The class SARule serves as a base class for IKERule and IPsecRule. Even though the class is concrete, it MUST not be instantiated. It defines a common connection point for associations to conditions and actions for both types of rules. Each SARule within a given IPsecPolicyGroup must contain a unique priority. Through its derivation from PolicyRule, an SARule (and therefore IKERule and IPsecRule) also has the PolicyRuleValidityPeriod association. The class definition for SARule is as follows:

```
NAME         SARule
DESCRIPTION  A base class for IKERule and IPsecRule.
DERIVED FROM PolicyRule (see [PCIM])
ABSTRACT     FALSE
PROPERTIES   PolicyRuleName (from PolicyRule)
             Enabled (from PolicyRule)
             ConditionListType (from PolicyRule)
             Priority (from PolicyRule)
             PolicyRoles (from PolicyRule)
```

4.3. The Class IKERule

The class IKERule associates Conditions and Actions for IKE phase 1 negotiations. The class definition for IKERule is as follows:

```
NAME         IKERule
DESCRIPTION  Associates Conditions and Actions for IKE phase 1
             negotiations.
DERIVED FROM SARule
ABSTRACT     FALSE
PROPERTIES   same as SARule
```

4.4. The Class IPsecRule

The class IPsecRule associates Conditions and Actions for IKE phase 2 negotiations for the IPsec DOI. The class definition for IPsecRule is as follows:

```
   NAME         IKERule
   DESCRIPTION  Associates Conditions and Actions for IKE phase 2
                negotiations for the IPsec DOI.
   DERIVED FROM SARule
```

ABSTRACT	FALSE
PROPERTIES	same as SARule

4.5. The Aggregation Class IPsecPolicyGroupInPolicyGroup

The class IPsecPolicyGroupInPolicyGroup allows multiple IPsec policies to be combined to into one effective policy. When merging policies, rule priorities are used in conjunction with the rule override point values to determine insertion points and for rule priority renumbering (if necessary to maintain uniqueness). The class definition for IPsecPolicyGroupInPolicyGroup is as follows:

NAME	IPsecPolicyGroupInPolicyGroup
DESCRIPTION	Associates a nested IPsecPolicyGroup with the IPsecPolicyGroup that contains it.
DERIVED FROM	PolicyGroupInPolicyGroup (see [PCIM])
ABSTRACT	FALSE
PROPERTIES	ContainingGroup[ref IPsecPolicyGroup[0..n]] ContainedGroup[ref IPsecPolicyGroup[0..n]] Precedence

4.5.1. The Reference ContainingGroup

The property ContainingGroup is inherited from PolicyGroupInPolicyGroup and is overridden to contain object reference to an IPsecPolicyGroup that contains one or more IPsecPolicyGroups. The [0..n] cardinality indicates that there may be zero or more IPsecPolicyGroups that contain any given IPsecPolicyGroup.

4.5.2. The Reference ContainedGroup

The property ContainedGroup is inherited from PolicyGroupInPolicyGroup and is overridden to contain an object reference to an IPsecPolicyGroup contained by one or more IPsecPolicyGroups. The [0..n] cardinality indicates that an IPsecPolicyGroup may contain zero or more IPsecPolicyGroups.

4.5.3. The Property Precedence

The property Precedence specifies the merge ordering of the nested IPsecPolicyGroups. The property is defined as follows:

NAME	Precedence
DESCRIPTION	Specifies the merge ordering of the nested IPsecPolicyGroups.
SYNTAX	unsigned 16-bit integer
VALUE	Any value between 1 and 2^16-1 inclusive. Lower values have higher precedence (i.e., 1 is the highest

precedence). The merging order of two ContainedGroups with the same precedence is undefined.

4.6. The Composition Class RuleForIKENegotiation

The class RuleForIKENegotiation associates an IKERule with the IPsecPolicyGroup that contains it. The class definition for RuleForIKENegotiation is as follows:

NAME	RuleForIKENegotiation
DESCRIPTION	Associates an IKERule with the IPsecPolicyGroup that contains it.
ABSTRACT	FALSE
PROPERTIES	ContainingGroup [ref IPsecPolicyGroup [1..1]]
	ContainedRule [ref IKERule [0..n]]

4.6.1. The Reference ContainingGroup

The property ContainingGroup contains an object reference to an IPsecPolicyGroup that contains one or more IKERules. The [1..1] cardinality indicates that an IKERule may be contained in only one IPsecPolicyGroup (i.e., IKERules are not shared across IPsecPolicyGroups).

4.6.2. The Reference ContainedRule

The property ContainedRule contains an object reference to an IKERule contained by an IPsecPolicyGroup. The [0..n] cardinality indicates that an IPsecPolicyGroup may contain zero or more IKERules.

4.7. The Composition Class RuleForIPsecNegotiation

The class RuleForIPsecNegotiation associates an IPsecRule with the IPsecPolicyGroup that contains it. The class definition for RuleForIPsecNegotiation is as follows:

NAME	RuleForIPsecNegotiation
DESCRIPTION	Associates an IPsecRule with the IPsecPolicyGroup that contains it.
ABSTRACT	FALSE
PROPERTIES	ContainingGroup [ref IPsecPolicyGroup [1..1]]
	ContainedRule [ref IPsecRule [0..n]]

4.7.1. The Reference ContainingGroup

The property ContainingGroup contains an object reference to an IPsecPolicyGroup that contains one or more IPsecRules. The [1..1] cardinality indicates that an IPsecRule may be contained in only one IPsecPolicyGroup (i.e., IPsecRules are not shared across IPsecPolicyGroups).

4.7.2. The Reference ContainedRule

The property ContainedRule contains an object reference to an IPsecRule contained by an IPsecPolicyGroup. The [0..n] cardinality

indicates that an IPsecPolicyGroup may contain zero or more IPsecRules.

4.8. The Aggregation Class SAConditionInRule

The class SAConditionInRule associates an SARule with the SACondition instances that trigger it. See [PCIM] for the usage for the properties GroupNumber and ConditionNegated. The class definition for SAConditionInRule is as follows:

NAME	SAConditionInRule
DESCRIPTION	Associates an SARule with the SACondition instances that trigger it.
DERIVED FROM	PolicyConditionInPolicyRule (see [PCIM])
ABSTRACT	FALSE
PROPERTIES	ContainingRule [ref SARule [0..n]] ContainedCondition [ref SACondition [0..n]] GroupNumber (from PolicyConditionInPolicyRule) ConditionNegated (from PolicyConditionInPolicyRule) SequenceNumber

4.8.1. The Reference ContainingRule

The property ContainingRule is inherited from PolicyConditionInPolicyRule and is overridden to contain an object reference to an SARule that contains one or more SAConditions. The [0..n] cardinality indicates that an SACondition may be contained in zero or more SARules.

4.8.2. The Reference ContainedCondition

The property ContainedCondition is inherited from PolicyConditionInPolicyRule and is overridden to contain an object reference to an SACondition that is contained by an SARule. The [0..n] cardinality indicates that an SARule may contain zero or more SAConditions.

4.8.3. The Property SequenceNumber

The property SequenceNumber specifies, for a given rule, the order in which the SACondition instances will be evaluated. The property is defined as follows:

NAME	SequenceNumber
DESCRIPTION	Specifies the evaluation order of the SAConditions.
SYNTAX	unsigned 16-bit integer
VALUE	Lower valued SAConditions are evaluated first. The order of evaluation of ContainedConditions with the same SequenceNumber value is undefined.

4.9. The Aggregation Class SAActionInRule

The SAActionInRule class associates an SARule with its primary SAAction. The class definition for SAActionInRule is as follows:

NAME	SAActionInRule
DESCRIPTION	Associates an SARule with its primary SAAction.
DERIVED FROM	PolicyActionInPolicyRule (see [PCIM])
ABSTRACT	FALSE
PROPERTIES	ContainingRule [ref SARule [0..n]]
	ContainedAction [ref SAAction [1..1]]

4.9.1. The Reference ContainingRule

The property ContainingRule is inherited from PolicyActionInPolicyRule and is overridden to contain an object reference to an SARule that contains an SAAction. The [0..n] cardinality indicates that an SAAction may be contained in zero or more SARules.

4.9.2. The Reference ContainedAction

The property ContainedAction is inherited from PolicyActionInPolicyRule and is overridden to contain an object reference to an SAAction that is contained by an SARule. The [1..1] cardinality indicates that an SARule may contain only one SAAction.

4.10. The Aggregation Class FallbackSAActionInRule

The class FallbackSAActionInRule associates an SARule with its ordered set of fallback actions. Fallback actions allow an administrator to define what action is to be take if the SAAction referenced by SAActionInRule fails for any reason. The class definition for FallbackSAActionInRule is as follows:

NAME	FallbackSAActionInRule
DESCRIPTION	Associates an SARule with the ordered set of fallback actions that should be attempted/applied in the case of failure of the primary SAAction.
ABSTRACT	FALSE
PROPERTIES	ContainingRule [ref SARule [0..n]]
	ContaintedAction [ref SAAction [0..n]]
	SequenceNumber

4.10.1. The Reference ContainingRule

The property ContainingRule contains an object reference to an SARule that contains one or more fallback SAActions. The [0..n] cardinality indicates that an fallback SAAction may be contained in zero or more SARules.

4.10.2. The Reference ContainedAction

The property ContainedAction contains an object reference to a fallback SAAction that is contained by one or more SARules. The

[0..n] cardinality indicates that an SARule may contain zero or more fallback SAActions.

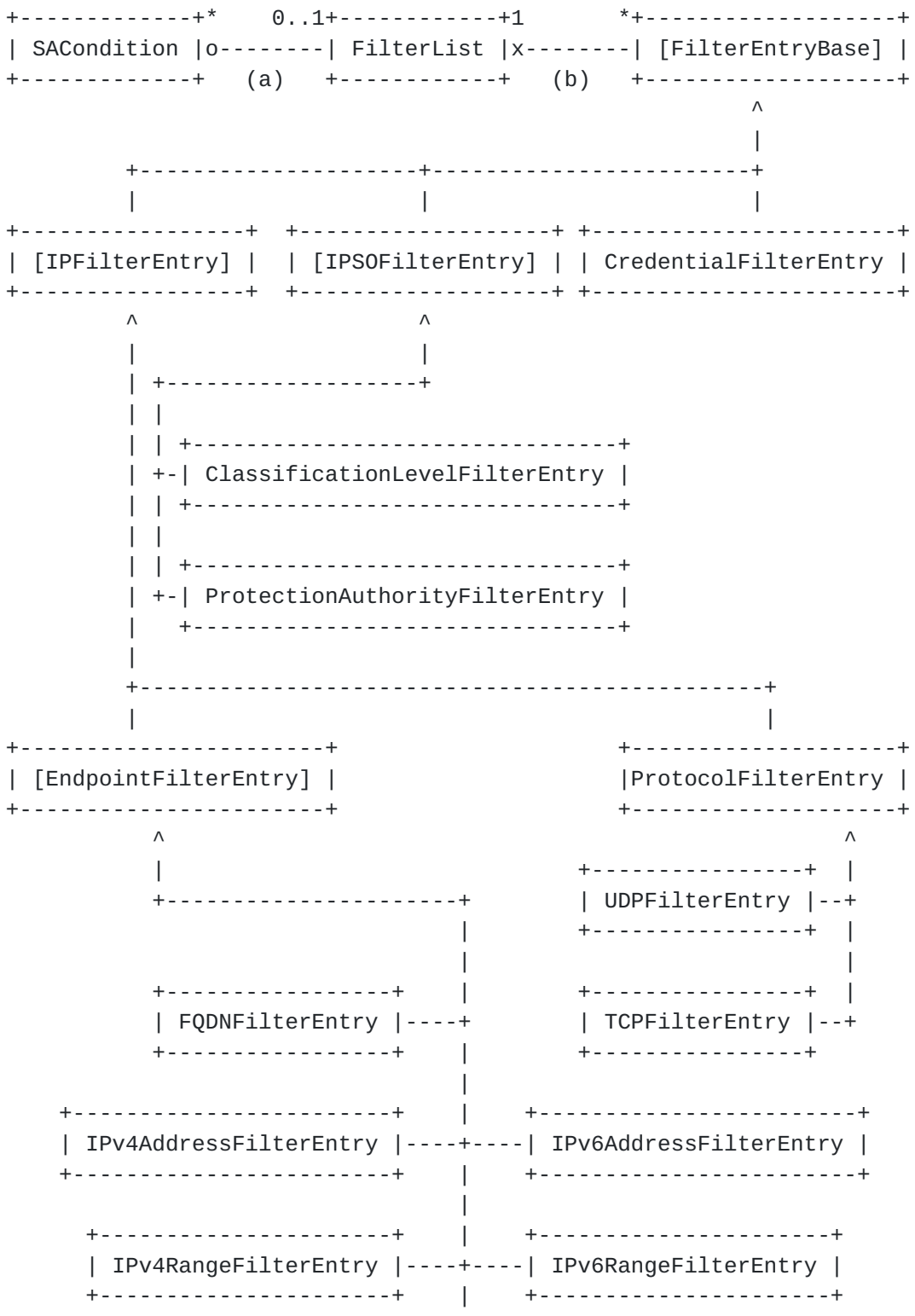
4.10.3. The Property SequenceNumber

The property SequenceNumber specifies, for a given rule, the order in which the fallback SAActions should be attempted. Once a fallback SAAction is successfully applied, then subsequent fallback SAActions should be ignored. The property is defined as follows:

NAME	SequenceNumber
DESCRIPTION	Specifies the order of attempted application for the fallback SAAction.
SYNTAX	unsigned 16-bit integer
VALUE	Lower valued fallback SAActions are attempted first. The order of attempt of ContainedActions with the same SequenceNumber value is undefined.

5. Condition and Filter Classes

The IPsec condition and filter classes are used to build the "if" part of the IKE and IPsec rules.




IPv4SubnetFilterEntry
IPv6SubnetFilterEntry

(a) FilterOfSACondition
(b) EntriesInFilterList

5.1. The Class SACondition

The class SACondition defines the preconditions for IKE and IPsec negotiations. The class definition for SACondition is as follows:

NAME	SACondition
DESCRIPTION	Defines the preconditions for IKE and IPsec negotiations.
DERIVED FROM	PolicyCondition (see [PCIM])
ABSTRACT	FALSE
PROPERTIES	PolicyConditionName (from PolicyCondition) StartupCondition

5.1.1. The Property StartupCondition

This property specifies the triggering event that caused the rule evaluation. The property is defined as follows:

NAME	StartupCondition
DESCRIPTION	Specifies the triggering event that cause the rule to be evaluated.
SYNTAX	unsigned 16-bit integer
VALUE	1 (OnBoot) - the rule is triggered after system boot. The FilterList associated with the SACondition contains the information that will be used to build the selectors. 2 (OnManual) - the rule is triggered manually in response to user input. The FilterList associated with the SACondition contains the information that will be used to build the selectors. 3 (OnDataTraffic) - the rule is triggered when packets without associated security associations are sent or received (traffic directionality is indicated by the Direction field of the associated FilterList). 4 (OnIKEMessage) - the rule is triggered when an incoming request for IKE negotiation is received.

5.2. The Class FilterList

The class FilterList aggregates an ANDed set of filters that are used for determining when an SACondition evaluates to true and therefore its associated SAAction should be performed. The class definition for FilterList is as follows:

NAME	FilterList
DESCRIPTION	Aggregates a set of filters for condition matching.

```
   ABSTRACT     FALSE
   PROPERTIES   Name
                Direction
```

5.2.1. The Property Name

This property specifies a user-friendly name for the FilterList. The property is defined as follows:

```
NAME         Name
DESCRIPTION  Specifies the user-friendly name for the FilterList.
SYNTAX       string
```

5.2.2. The Property Direction

This property specifies whether or the FilterList will be used on incoming, outgoing, or bi-directional traffic. Direction is only useful for filter types that inspect traffic parameters and when the StartupCondition property in the SACondition is set to OnDataTraffic (3). The property is defined as follows:

```
NAME         Direction
DESCRIPTION  Specifies what kind of traffic will be checked -
             incoming, outgoing, or bi-directional.
SYNTAX       unsigned 16-bit integer
VALUE        1 - Incoming
             2 - Outgoing
             3 - Bi-directional
```

5.3. The Abstract Class FilterEntryBase

The abstract class FilterEntryBase serves as the base class for the specific filter class. The class definition for FilterEntryBase is as follows:

```
NAME         FilterEntryBase
DESCRIPTION  Serves as the base class for specific filter classes.
ABSTRACT     TRUE
PROPERTIES   Name
             IsNegated
```

5.3.1. The Property Name

This property specifies a user-friendly name for the filter. The property is defined as follows:

```
NAME         Name
DESCRIPTION  Specifies the user-friendly name for the filter.
SYNTAX       string
```

5.3.2. The Property IsNegated

This property specifies whether or not the result of the boolean

result of the filter evaluation should be negated. The property is defined as follows:

NAME IsNegated

DESCRIPTION	Specifies whether or not to negate the result of the evaluation of the filter.
SYNTAX	boolean
VALUE	A value of true means that the boolean result of the filter evaluation of the filter will be negated. A value of false means that the boolean result of the evaluation of the filter will not be altered.

5.4. The Abstract Class IPFilterEntry

The abstract class IPFilterEntry serves as a base class for filter entries which are used to match against the 5-tuple (i.e., source and destination address, protocol, and source and destination port) information in the IP packet. The class definition for IPFilterEntry is as follows:

NAME	IPFilterEntry
DESCRIPTION	Serves as the base class for IP 5-tuple filters.
DERIVED FROM	FilterEntryBase
ABSTRACT	TRUE

5.5. The Abstract Class EndpointFilterEntry

The abstract class EndpointFilterEntry serves as a base class for filters which match against IP addresses (source or destination). The class definition for EndpointFilterEntry is as follows:

NAME	EndpointFilterEntry
DESCRIPTION	Serves as the base class for filters which match against IP addresses.
DERIVED FROM	IPFilterEntry
ABSTRACT	TRUE
PROPERTIES	ApplyToDestination

5.5.1. The Property ApplyToDestination

This property specifies whether or not the address to test against is the source or the destination IP address. The property is defined as follows:

NAME	ApplyToDestination
DESCRIPTION	Specifies which IP address to test, source or destination.
SYNTAX	boolean
VALUE	A value of true means that the destination IP address should be tested against. A value of false means that the source IP address should be tested against.

5.6. The Class IPv4AddressFilterEntry

The class IPv4AddressFilterEntry specifies a filter that will match against a single IPv4 address. The class definition for IPv4AddressFilterEntry is as follows:

```
NAME         IPv4AddressFilterEntry
DESCRIPTION  Defines the match filter for an IPv4 address.
DERIVED FROM EndpointFilterEntry
ABSTRACT     FALSE
PROPERTIES   Address
```

5.6.1. The Property Address

This property specifies the IPv4 address that will be used in the equality test. The property is defined as follows:

```
NAME         Address
DESCRIPTION  Specifies the IPv4 address to match against.
SYNTAX       unsigned 32-bit integer
```

5.7. The Class IPv4RangeFilterEntry

The class IPv4RangeFilterEntry specifies a filter for testing if an IPv4 address is between the start address and end address inclusively. The class definition for IPv4RangeFilterEntry is as follows:

```
NAME         IPv4RangeFilterEntry
DESCRIPTION  Defines the match filter for an IPv4 address range.
DERIVED FROM EndpointFilterEntry
ABSTRACT     FALSE
PROPERTIES   StartAddress
             EndAddress
```

5.7.1. The Property StartAddress

This property specifies the first IPv4 address in the address range. The property is defined as follows:

```
NAME         StartAddress
DESCRIPTION  Specifies the start of the IPv4 address range.
SYNTAX       unsigned 32-bit integer
```

5.7.2. The Property EndAddress

This property specifies the last IPv4 address in the address range. The property is defined as follows:

```
NAME         EndAddress
DESCRIPTION  Specifies the end of the IPv4 address.
SYNTAX       unsigned 32-bit integer
VALUE        EndAddress must be greater than or equal to
             StartAddress.
```

5.8. The Class IPv4SubnetFilterEntry

The class IPv4SubnetFilterEntry specifies a filter for testing if an IPv4 address is in the specified subnet. The class definition for IPv4SubnetFilterEntry is as follows:

```
NAME          IPv4SubnetFilterEntry
DESCRIPTION   Defines the match filter for an IPv4 subnet.
DERIVED FROM  EndpointFilterEntry
ABSTRACT      FALSE
PROPERTIES    Address
              Mask
```

5.8.1. The Property Address

This property specifies the IPv4 subnet. The property is defined as follows:

```
NAME          Address
DESCRIPTION   Specifies the IPv4 subnet.
SYNTAX        unsigned 32-bit integer
```

5.8.2. The Property Mask

This property specifies the IPv4 mask. The property is defined as follows:

```
NAME          Mask
DESCRIPTION   Specifies the IPv4 mask.
SYNTAX        unsigned 32-bit integer
VALUE         A special value of 0.0.0.0, coupled with an Address
              value of 0.0.0.0 can be used to specify all addresses.
```

5.9. The Class IPv6AddressFilterEntry

The class IPv6AddressFilterEntry specifies a filter that will match against a single IPv6 address. The class definition for IPv6AddressFilterEntry is as follows:

```
NAME          IPv6AddressFilterEntry
DESCRIPTION   Defines the match filter for an IPv4 address.
DERIVED FROM  EndpointFilterEntry
ABSTRACT      FALSE
PROPERTIES    Address
```

5.9.1. The Property Address

This property specifies the IPv6 address that will be used in the equality test. The property is defined as follows:

```
NAME          Address
```

DESCRIPTION Specifies the IPv6 address to match against.
SYNTAX   byte[16]

5.10. The Class IPv6RangeFilterEntry

The class IPv6RangeFilterEntry specifies a filter for testing if an IPv6 address is between the start address and end address inclusively. The class definition for IPv6RangeFilterEntry is as follows:

```
NAME          IPv6RangeFilterEntry
DESCRIPTION   Defines the match filter for an IPv6 address range.
DERIVED FROM  EndpointFilterEntry
ABSTRACT      FALSE
PROPERTIES    StartAddress
              EndAddress
```

5.10.1. The Property StartAddress

This property specifies the first IPv6 address in the address range. The property is defined as follows:

```
NAME          StartAddress
DESCRIPTION   Specifies the start of the IPv6 address range.
SYNTAX        byte[16]
```

5.10.2. The Property EndAddress

This property specifies the last IPv6 address in the address range. The property is defined as follows:

```
NAME          EndAddress
DESCRIPTION   Specifies the end of the IPv6 address.
SYNTAX        byte[16]
VALUE         EndAddress must be greater than or equal to
              StartAddress.
```

5.11. The Class IPv6SubnetFilterEntry

The class IPv6SubnetFilterEntry specifies a filter for testing if an IPv6 address is in the specified subnet. The class definition for IPv4SubnetFilterEntry is as follows:

```
NAME          IPv6SubnetFilterEntry
DESCRIPTION   Defines the match filter for an IPv6 subnet.
DERIVED FROM  EndpointFilterEntry
ABSTRACT      FALSE
PROPERTIES    Address
              Mask
```

5.11.1. The Property Address

This property specifies the IPv6 subnet. The property is defined as

follows:

```
NAME         Address
DESCRIPTION  Specifies the IPv6 subnet.
```

```
SYNTAX       byte[16]
```

5.11.2. The Property Mask

This property specifies the IPv6 mask. The property is defined as follows:

```
NAME         Mask
DESCRIPTION  Specifies the IPv6 mask.
SYNTAX       byte[16]
VALUE        A special value of 0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0,
             coupled with an Address value of
             0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0 can be used to specify
             all addresses.
```

5.12. The Class FQDNFilterEntry

The class FQDNFilterEntry specifies a filter for mathcing against a single or wild-carded DNS name. The class definition for FQDNFilterEntry is as follows:

```
NAME         FQDNFilterEntry
DESCRIPTION  Defines the match filter for a DNS name.
DERIVED FROM EndpointFilterEntry
ABSTRACT     FALSE
PROPERTIES   Name
```

5.12.1. The Property Name

This property specifies the DNS name to match against. The property is defined as follows:

```
NAME         Address
DESCRIPTION  Specifies the DNS name.
SYNTAX       string
VALUE        The DNS name can be fully qualified (for example,
             foo.intel.com) or partially qualified (*.intel.com).
```

5.13. The Class ProtocolFilterEntry

The class ProtocolFilterEntry specifies a filter for testing against an IP protocol. The class definition for ProtocolFilterEntry is as follows:

```
NAME         ProtocolFilterEntry
DESCRIPTION  Defines a match filter for IP protocol.
DERIVED FROM IPFilterEntry
ABSTRACT     FALSE
PROPERTIES   Protocol
```

5.13.1. The Property Protocol

This property specifies the IP protocol to match against. The property is defined as follows:

NAME	Protocol
DESCRIPTION	Specifies the IP protocol.
SYNTAX	unsigned 8-bit integer
VALUE	A value of zero matches against any protocol. Any other value is the IP protocol number.

5.14. The Class UDPFilterEntry

The class UDPFilterEntry specifies a filter for testing if a UDP port is between the start port and end port inclusively. It is assumed that the Protocol property from the ProtocolFilterEntry class will contain the value 17 (i.e., UDP). The class definition for UDPFilterEntry is as follows:

NAME	UDPFilterEntry
DESCRIPTION	Defines the match filter for a UDP port range.
DERIVED FROM	ProtocolFilterEntry
ABSTRACT	FALSE
PROPERTIES	StartPort
	EndPort

5.14.1. The Property StartPort

This property specifies the first port in the UDP port range. The property is defined as follows:

NAME	StartPort
DESCRIPTION	Specifies the start of the UDP port range.
SYNTAX	unsigned 16-bit integer

5.14.2. The Property EndPort

This property specifies the last port in the UDP port range. The property is defined as follows:

NAME	EndPort
DESCRIPTION	Specifies the end of the UDP port range.
SYNTAX	unsigned 16-bit integer
VALUE	EndPort must be greater than or equal to StartPort.

5.15. The Class TCPFilterEntry

The class TCPFilterEntry specifies a filter for testing if a TCP port is between the start port and end port inclusively. It is assumed that the Protocol property from the ProtocolFilterEntry class will contain the value 6 (i.e., TCP). The class definition

for TCPFilterEntry is as follows:

NAME TCPFilterEntry
DESCRIPTION Defines the match filter for a TCP port range.

```
DERIVED FROM ProtocolFilterEntry
ABSTRACT     FALSE
PROPERTIES   StartPort
             EndPort
```

5.15.1. The Property StartPort

This property specifies the first port in the TCP port range. The property is defined as follows:

```
NAME         StartPort
DESCRIPTION  Specifies the start of the TCP port range.
SYNTAX       unsigned 16-bit integer
```

5.15.2. The Property EndPort

This property specifies the last port in the TCP port range. The property is defined as follows:

```
NAME         EndPort
DESCRIPTION  Specifies the end of the TCP port range.
SYNTAX       unsigned 16-bit integer
VALUE        EndPort must be greater than or equal to StartPort.
```

5.16. The Abstract Class IPSOFilterEntry

The abstract class IPSOFilterEntry serves as a base class for the IP Security Option (IPSO) filters. The class definition for IPSOFilterEntry is as follows:

```
NAME         IPSOFilterEntry
DESCRIPTION  Serves as the base class for the IPSO filters.
DERIVED FROM FilterEntryBase
ABSTRACT     TRUE
```

5.17. The Class ClassificationLevelFilterEntry

The class ClassificationLevelFilterEntry specifies a filter for matching against the classification level IPSO field type. The class definition for ClassificationLevelFilterEntry is as follows:

```
NAME         ClassificationLevelFilterEntry
DESCRIPTION  Defines the filter for the IPSO classification level.
DERIVED FROM IPSOFilterEntry
ABSTRACT     FALSE
PROPERTIES   Level
```

5.17.1. The Property Level

This property specifies the classification level to match against. The property is defined as follows:

NAME Level

```
DESCRIPTION  Specifies the classification level.
SYNTAX       unsigned 16-bit integer
VALUE        61 - Top Secret
             90 - Secret
             150 - Confidential
             171 - Unclassified
```

5.18. The Class ProtectionAuthorityFilterEntry

The class ProtectionAuthorityFilterEntry specifies a filter for matching against the protection authority IPSO field type. The class definition for ProtectionAuthorityFilterEntry is as follows:

```
NAME         ProtectionAuthorityFilterEntry
DESCRIPTION  Defines the filter for the IPSO protection authority.
DERIVED FROM IPSOFilterEntry
ABSTRACT     FALSE
PROPERTIES   Authority
```

5.18.1. The Property Authority

This property specifies the protection authority to match against. The property is defined as follows:

```
NAME         Authority
DESCRIPTION  Specifies the protection authority.
SYNTAX       unsigned 16-bit integer
VALUE        0 - GENSER
             1 - SIOP-ESI
             2 - SCI
             3 - NSA
             4 - DOE
```

5.19. The Class CredentialFilterEntry

The class CredentialFilterEntry defines a filter for matching against credential information that was obtained during the IKE phase 1 negotiation. This information can be identity information (such as User FQDN) or information retrieved from credential information (for example, fields from a certificate). This information can be used as a form of access control. The class definition for CredentialFilterEntry is as follows:

```
NAME         CredentialFilterEntry
DESCRIPTION  Defines the filter for matching against IKE phase 1
             credential/identity information.
DERIVED FROM FilterBaseEntry
ABSTRACT     FALSE
PROPERTIES   To Be Determined...
```

5.20. The Aggregation Class FilterOfSACondition

The class FilterOfSACondition associates an SACondition with the filter specifications (FilterList) that make up the condition. The class definition for FilterOfSACondition is as follows:

```
NAME         FilterOfSACondition
DESCRIPTION  Associates a condition with the filter list that make
             up the individual condition elements.
ABSTRACT     FALSE
PROPERTIES   Antecedent [ref FilterList[0..1]]
             Dependent [ref SACondition [0..n]]
```

5.20.1. The Reference Antecedent

The property Antecedent contains an object reference to a FilterList that is contained in one or more SAConditions. The [0..1] cardinality indicates that an SACondition may have zero or one FilterList.

5.20.2. The Reference Dependent

The property Dependent contains an object reference to an SACondition that contains an FilterList. The [0..n] cardinality indicates that a FilterList may be contained in zero or more SAConditions.

5.21. The Composition Class EntriesInFilterList

The class EntriesInFilterList associates the individual FilterEntryBases with a FilterList. Together these individual FilterEntryBases can create complex conditions. The class definition for EntriesInFilterList is as follows:

```
NAME         EntriesInFilterList
DESCRIPTION  Associates a FilterList with the set of individual
             filters.
ABSTRACT     FALSE
PROPERTIES   Antecedent [ref FilterEntryBase[0..n]]
             Dependent [ref FilterList [1..1]]
             EntrySequence
```

5.21.1. The Reference Antecedent

The property Antecedent contains an object reference to a FilterEntryBase that is contained in a FilterList. The [0..n] cardinality indicates that a FilterList may have zero or more FilterEntryBases.

5.21.2. The Reference Dependent

The property Dependent contains an object reference to a FilterList that contains zero or more FilterEntryBases. The [1..1] cardinality indicates that a FilterEntryBase may be contained in one and only

one FilterLists (i.e., FilterEntryBases cannot be shared between FilterLists).

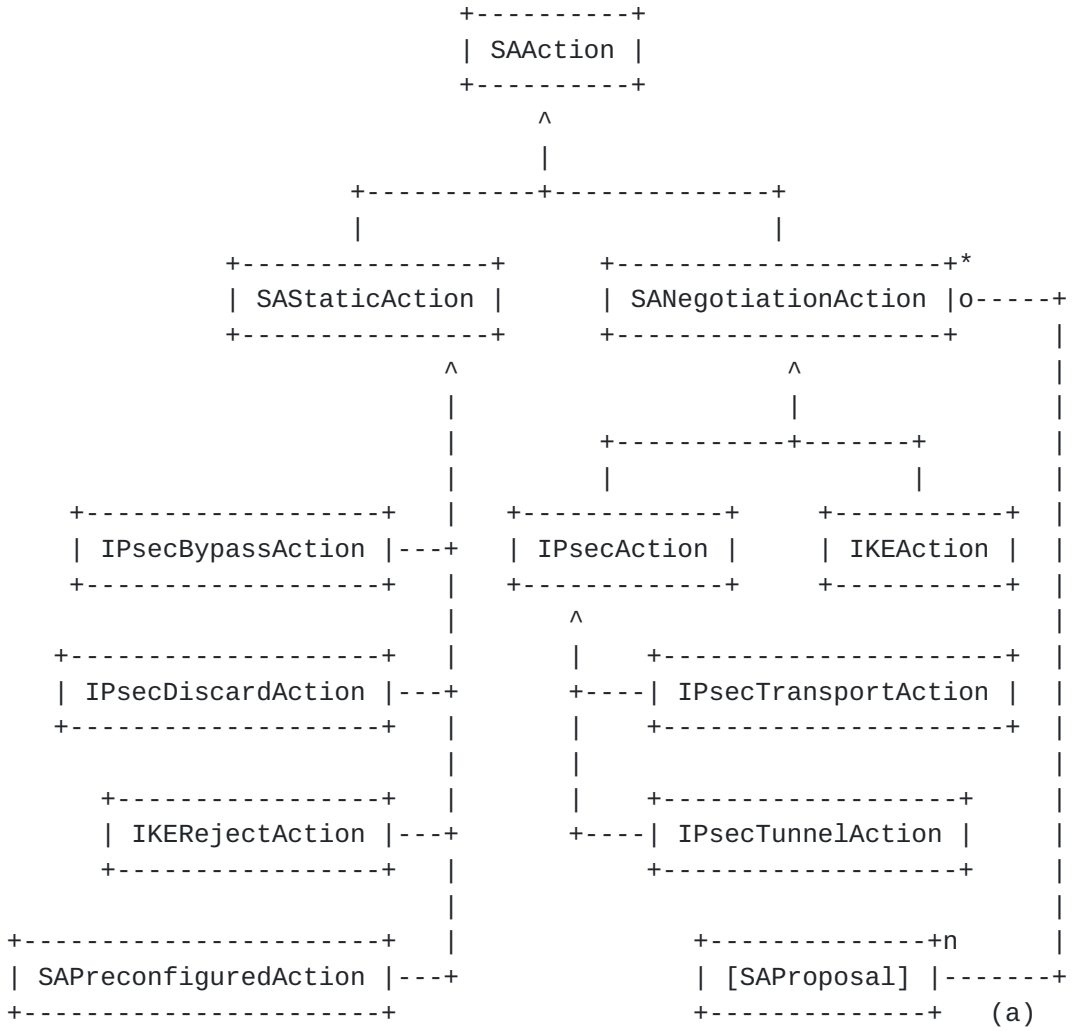
5.21.3. The Property EntrySequence

The property EntrySequence specifies, for a given FilterList, the order in which the filters should be checked. The property is defined as follows:

NAME	EntrySequence
DESCRIPTION	Specifies the order to check the filters in a FilterList.
SYNTAX	unsigned 16-bit integer
VALUE	Lower valued filters are checked first. The order of checking of FilterEntryBases with the same EntrySequence value is undefined.

6. Action Classes

The action classes are used to model the different actions an IPsec device may take when the evaluation of the associated condition results in a match.



(a) ContainedProposal

6.1. The Class SAAction

The class SAAction serves as the base class for IKE and IPsec actions. Although the class is concrete, it MUST not be instantiated. The class definition for SAAction is as follows:

NAME SAAction
DESCRIPTION The base class for IKE and IPsec actions.
DERIVED FROM PolicyAction (see [PCIM])
ABSTRACT FALSE

PROPERTIES PolicyActionName (from PolicyAction)

6.2. The Class SAStaticAction

The class SAStaticAction serves as the base class for IKE and IPsec actions that do not require any negotation. Although the class is concrete, it MUST not be instantiated. The class definition for SAStaticAction is as follows:

NAME	SAStaticAction
DESCRIPTION	The base class for IKE and IPsec actions that do not require any negotiation.
DERIVED FROM	SAAction
ABSTRACT	FALSE
PROPERTIES	LifetimeSeconds

6.2.1. The Property LifetimeSeconds

The property LifetimeSeconds specifies how long the security association derived from this action should be used. The property is defined as follows:

NAME	LifetimeSeconds
DESCRIPTION	Specifies the amount of time (in seconds) that a security association derived from this action should be used.
SYNTAX	unsigned 32-bit integer
VALUE	A value of zero indicates that there is not a lifetime associated with this action (i.e., infinite lifetime). A nono-zero value is typically used in conjunction with fallback actions performed when there is a negotiation failure of some sort.

6.3. The Class IPsecBypassAction

The class IPsecBypassAction is used when packets are allowed to be processed without applying IPsec to them. This is the same as stating that packets are allowed to flow in the clear. The class definition for IPsecBypassAction is as follows:

NAME	IPsecBypassAction
DESCRIPTION	Specifies that packets are to be allowed to pass in the clear.
DERIVED FROM	SAStaticAction
ABSTRACT	FALSE

6.4. The Class IPsecDiscardAction

The class IPsecDiscardAction is used when packets are to be discarded. This is the same as stating that packets are to be denied. The class definition for IPsecDiscardAction is as follows:

NAME	IPsecDiscardAction

```
   DESCRIPTION  Specifies that packets are to be discarded.
   DERIVED FROM SAStaticAction
   ABSTRACT     FALSE
   PROPERTIES   DoLogging
```

6.4.1. The Property DoLogging

The property DoLogging specifies whether or not an audit message should be logged when a packet is discarded. The property is defined as follows:

NAME	DoLogging
DESCRIPTION	Specifies if an audit message should be logged when a packet is discarded.
SYNTAX	boolean
VALUE	A value of true indicates that logging should be done for this action. A value of false indicates logging should not be done for this action.

6.5. The Class IKERejectAction

The class IKERejectAction is used to prevent attempting an IKE negotiation with the peer(s). The class definition for IKERejectAction is as follows:

NAME	IKERejectAction
DESCRIPTION	Specifies that an IKE negotiation should not even be attempted.
DERIVED FROM	SAStaticAction
ABSTRACT	FALSE
PROPERTIES	DoLogging

6.5.1. The Property DoLogging

The property DoLogging specifies whether or not an audit message should be logged when a determination is made to prevent an IKE negotiation. The property is defined as follows:

NAME	DoLogging
DESCRIPTION	Specifies if an audit message should be logged when IKE negotiation is prohibited.
SYNTAX	boolean
VALUE	A value of true indicates that logging should be done for this action. A value of false indicates logging should not be done for this action.

6.6. The Class SAPreconfiguredAction

The class SAPreconfiguredAction is used to create a security association using preconfigured, hard-wired algorithms and keys. The class definition for SAPreconfiguredAction is as follows:

NAME	SAPreconfiguredAction

```
   DESCRIPTION  Specifies preconfigured algorithm and keying
                information for creation of a security association.
   DERIVED FROM SAStaticAction
   ABSTRACT     FALSE
```

PROPERTIES To Be Determined...

6.7. The Class SANegotiationAction

The class SANegotiationAction serves as the base class for IKE and IPsec actions which result in a IKE negotiation. Although the class is concrete, is MUST not be instantiated. The class definition for SANegotiationAction is as follows:

NAME	SANegotiationAction
DESCRIPTION	A base class for IKE and IPsec actions that specifies the parameters that are common for IKE phase 1 and IKE phase 2 IPsec DOI negotiations.
DERIVED FROM	SAAction
ABSTRACT	FALSE
PROPERTIES	MinLifetimeSeconds MinLifetimeKilobytes RefreshThresholdSeconds RefreshThresholdKilobytes IdleDurationSeconds

6.7.1. The Property MinLifetimeSeconds

The property MinLifetimeSeconds specifies the minimum seconds lifetime that will be accepted from the peer. MinLifetimeSeconds is used to prevent certain denial of service attacks where the peer requests an arbitrarily low lifetime value, causing renegotiations with correspondingly expensive Diffie-Hellman operations. The property is defined as follows:

NAME	MinLifetimeSeconds
DESCRIPTION	Specifies the minimum acceptable seconds lifetime.
SYNTAX	unsigned 32-bit integer
VALUE	A value of zero indicates that there is no minimum value. A non-zero value specifies the minimum seconds lifetime.

6.7.2. The Property MinLifetimeKilobytes

The property MinLifetimeKilobytes specifies the minimum kilobyte lifetime that will be accepted from the peer. MinLifetimeKilobytes is used to prevent certain denial of service attacks where the peer requests an arbitrarily low lifetime value, causing renegotiations with correspondingly expensive Diffie-Hellman operations. The property is defined as follows:

NAME	MinLifetimeKilobytes
DESCRIPTION	Specifies the minimum acceptable kilobyte lifetime.
SYNTAX	unsigned 32-bit integer

VALUE A value of zero indicates that there is no minimum value. A non-zero value specifies the minimum kilobyte lifetime.

6.7.3. The Property RefreshThresholdSeconds

The property RefreshThresholdSeconds specifies what percentage of the seconds lifetime can expire before IKE should attempt to renegotiate the IPsec security association. A random value may be added to the calculated threshold (percentage x seconds lifetime) to reduce the chance of both peers attempting to renegotiate at the same time. The property is defined as follows:

NAME	RefreshThresholdSeconds
DESCRIPTION	Specifies the percentage of seconds lifetime that has expired before the IPsec security association is renegotiated.
SYNTAX	unsigned 8-bit integer
VALUE	A value between 1 and 100 representing a percentage. A value of 100 indicates that the IPsec security association should not be renegotiated until the seconds lifetime has been reached.

6.7.4. The Property RefreshThresholdKilobytes

The property RefreshThresholdKilobytes specifies what percentage of the kilobyte lifetime can expire before IKE should attempt to renegotiate the IPsec security association. A random value may be added to the calculated threshold (percentage x kilobyte lifetime) to reduce the chance of both peers attempting to renegotiate at the same time. The property is defined as follows:

NAME	RefreshThresholdKilobytes
DESCRIPTION	Specifies the percentage of kilobyte lifetime that has expired before the IPsec security association is renegotiated.
SYNTAX	unsigned 8-bit integer
VALUE	A value between 1 and 100 representing a percentage. A value of 100 indicates that the IPsec security association should not be renegotiated until the kilobyte lifetime has been reached.

6.7.5. The Property IdleDurationSeconds

The property IdleDurationSeconds specifies how many seconds a security association may remain idle (i.e., no traffic protected using the security association) before it is deleted. The property is defined as follows:

NAME	IdleDurationSeconds
DESCRIPTION	Specifies how long, in seconds, a security association may remain unused before it is deleted.

```
   SYNTAX       unsigned 32-bit integer
   VALUE        A value of zero indicates that idle detection should
                not be used for the security association.  Any non-zero
```

value indicates the number of seconds the security association may remain unused.

6.8. The Class IPsecAction

The class IPsecAction serves as the base class for IPsec transport and tunnel actions. It specifies the parameters used for an IKE phase 2 IPsec DOI negotiation. Although the class is concrete, is MUST not be instantiated. The class definition for IPsecAction is as follows:

NAME	IPsecAction
DESCRIPTION	A base class for IPsec transport and tunnel actions that specifies the parameters for IKE phase 2 IPsec DOI negotiations.
DERIVED FROM	SANegotiationAction
ABSTRACT	FALSE
PROPERTIES	UsePFS UseIKEGroup GroupId Granularity

6.8.1. The Property UsePFS

The property UsePFS specifies whether or not perfect forward secrecy should be used when refreshing keys. The property is defined as follows:

NAME	UsePFS
DESCRIPTION	Specifies the whether or not to use PFS.
SYNTAX	boolean
VALUE	A value of true indicates that PFS should be used. A value of false indicates that PFS should not be used.

6.8.2. The Property UseIKEGroup

The property UseIKEGroup specifies whether or not phase 2 should use the same Diffie-Hellman as was used in phase 1. UseIKEGroup is ignored if UsePFS is false. The property is defined as follows:

NAME	UseIKEGroup
DESCRIPTION	Specifies whether or not to use the same GroupId for phase 2 as was used in phase 1. If UsePFS is false, then UseIKEGroup is ignored.
SYNTAX	boolean
VALUE	A value of true indicates that the phase 2 GroupId should be the same as phase 1. A value of false indicates that the property GroupId will contain the Diffie-Hellman group to use for phase 2.

6.8.3. The Property GroupId

The property GroupId specifies the Diffie-Hellman group to use for phase 2. GroupId is ignored if (1) the property UsePFS is false, or (2) the property UsePFS is true and the property UseIKEGroup is true. The property is defined as follows:

NAME	GroupId
DESCRIPTION	Specifies the Diffie-Hellman group to use for phase 2 when the property UsePFS is true and the property UseIKEGroup is false.
SYNTAX	unsigned 16-bit integer
VALUE	1 - 768-bit MODP group 2 - 1024-bit MODP group 3 - EC2N group on GP[2^155] 4 - EC2N group on GP[2^185] 5 - 1536-bit MODP group

6.8.4. The Property Granularity

The property Granularity specifies whether the proposed selector for the security association should be derived from the traffic that triggered the negotiation (Narrow) or from the FilterList of the Condition(s) that matched the rule (Wide). The property is defined as follows:

NAME	Granularity
DESCRIPTION	Specifies the how the proposed selector for the security association will be created.
SYNTAX	unsigned 8-bit integer
VALUE	1 - The selector is created by using the FilterList information from the condition that matched the traffic parameters. This is called a Wide selector as it could for instance contain a IP subnet or range. 2 - The selector is created by using the traffic parameters (i.e., the 5-tuple of the traffic). This is called a Narrow selector.

6.9. The Class IPsecTransportAction

The class IPsecTransportAction is a subclass of IPsecAction that is used to specify use of an IPsec transport mode security association. The class definition for IPsecTransportAction is as follows:

NAME	IPsecTransportAction
DESCRIPTION	Specifies that an IPsec transport mode security association should be negotiated.
DERIVED FROM	IPsecAction
ABSTRACT	FALSE

6.10. The Class IPsecTunnelAction

The class IPsecTunnelAction is a subclass of IPsecAction that is used to specify use of an IPsec tunnel mode security association. The class definition for IPsecTunnelAction is as follows:

```
NAME         IPsecTunnelAction
DESCRIPTION  Specifies that an IPsec tunnel mode security
             association should be negotiated.
DERIVED FROM IPsecAction
ABSTRACT     FALSE
PROPERTIES   PeerGateway
             DFHandling
```

6.10.1. The Property PeerGateway

The property PeerGateway specifies the IP address or DNS name of the peer gateway. The property is defined as follows:

```
NAME         PeerGateway
DESCRIPTION  Specifies peer gateway's IP address or DNS name.
SYNTAX       string
VALUE        Either (1) IPv4 address in dotted quad format, (2) IPv6
             address in ... format, or (3) a DNS name.
```

6.10.2. The Property DFHandling

The property DFHandling specifies how the Don't Fragment (DF) bit should be managed by the tunnel. The property is defined as follows:

```
NAME         DFHandling
DESCRIPTION  Specifies the DF bit is managed by the tunnel.
SYNTAX       unsigned 8-bit integer
VALUE        1 - DF bit is copied.
             2 - DF bit is set.
             3 - DF bit is cleared.
```

6.11. The Class IKEAction

The class IKEAction specifies the parameters that are to be used for IKE phase 1 negotiation. The class definition for IKEAction is as follows:

```
NAME         IKEAction
DESCRIPTION  Specifies the IKE phase 1 negotiation parameters.
DERIVED FROM SANegotiationAction
ABSTRACT     FALSE
PROPERTIES   RefreshThresholdDerivedKeys
             ExchangeMode
             UseIKEIdentityType
```

6.11.1. The Property RefreshThresholdDerivedKeys

The property RefreshThresholdDerivedKeys specifies what percentage of the derived key limit (see the LifetimeDerivedKeys property of IKEProposal) can expire before IKE should attempt to renegotiate the IKE phase 1 security association. A random value may be added to

the calculated threshold (percentage x derived key limit) to reduce the chance of both peers attempting to renegotiate at the same time. The property is defined as follows:

NAME	RefreshThresholdKilobytes
DESCRIPTION	Specifies the percentage of derived key limit that has expired before the IKE phase 1 security association is renegotiated.
SYNTAX	unsigned 8-bit integer
VALUE	A value between 1 and 100 representing a percentage. A value of 100 indicates that the IKE phase 1 security association should not be renegotiated until the derived key limit has been reached.

6.11.2. The Property ExchangeMode

The property ExchangeMode specifies which IKE mode should be used for IKE phase 1 key negotiations. The property is defined as follows:

NAME	ExchangeMode
DESCRIPTION	Specifies the IKE negotiation mode for phase 1.
SYNTAX	unsigned 16-bit integer
VALUE	1 - base mode 2 - main mode 4 - aggressive mode

6.11.3. The Property UseIKEIdentityType

The property UseIKEIdentityType specifies what IKE identity type should be used when negotiating with the peer. This information is used in conjunction the IKE identities available on the system. The property is defined as follows:

NAME	UseIKEIdentityType
DESCRIPTION	Specifies the IKE identity to use during negotiation.
SYNTAX	unsigned 16-bit integer
VALUE	1 - IPv4 Address 2 - FQDN 3 - User FQDN 4 - IPv4 Subnet 5 - IPv6 Address 6 - IPv6 Subnet 7 - IPv4 Address Range 8 - IPv6 Address Range 9 - DER-Encoded ASN.1 X.500 Distinguished Name 10 - DER-Encoded ASN.1 X.500 GeneralName 11 - Key ID

6.12. The Aggregation Class ContainedProposal

The class ContainedProposal associates an ordered list of SAProposals with the SANegotiationAction that contains it. If the

referenced SANegotiationAction object is an IKEAction, then the referenced SAProposal object must be an IKEProposal. If the referenced SANegotiationAction object is an IPsecTransportAction or an IPsecTunnelAction, then the referenced SAProposal object must be an IPsecProposal. The class definition for ContainedProposal is as follows:

NAME	ContainedProposal
DESCRIPTION	Associates an ordered list of SAProposals with an SANegotiationAction.
ABSTRACT	FALSE
PROPERTIES	GroupComponent[ref SANegotiationAction[0..n]] PartComponent[ref SAProposal[1..n]] SequenceNumber

6.12.1. The Reference GroupComponent

The property GroupComponent contains an object reference to an SANegotiationAction that contains one or more SAProposals. The [0..n] cardinality indicates that there may be zero or more SANegotiationActions that contain any given SAProposal.

6.12.2. The Reference PartComponent

The property PartComponent contains an object reference to an SAProposal contained by one or more SANegotiationActions. The [1..n] cardinality indicates that an SANegotiationAction MUST contain at least one SAProposal.

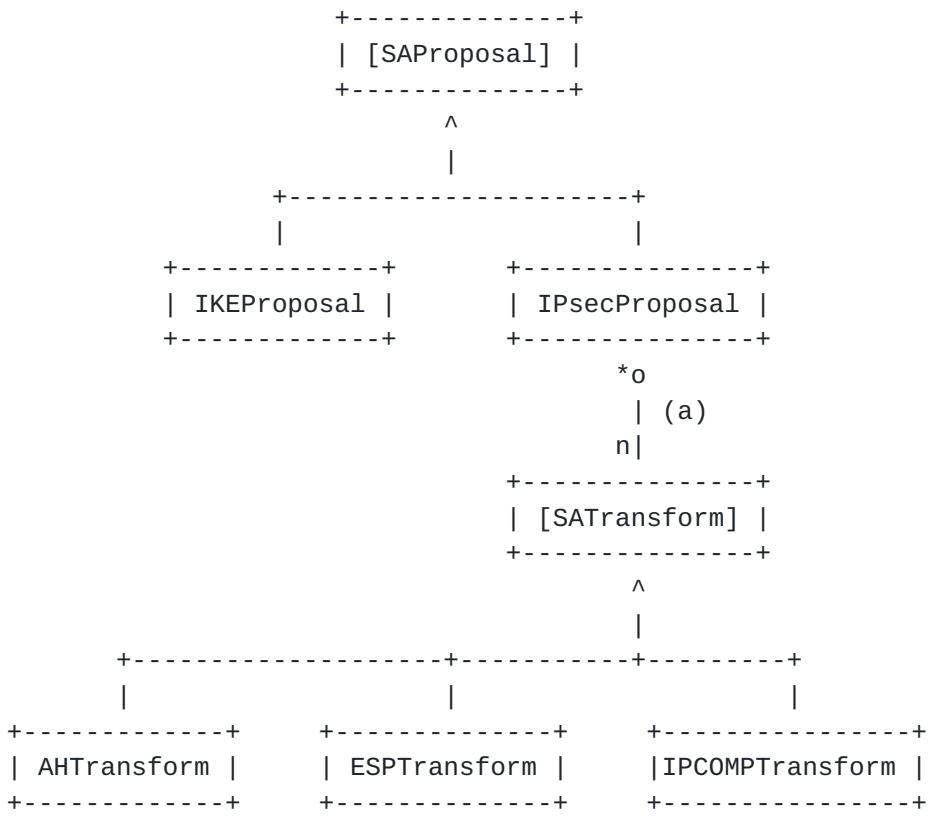
6.12.3. The Property SequenceNumber

The property SequenceNumber specifies the order of preference for the SAProposals. The property is defined as follows:

NAME	SequenceNumber
DESCRIPTION	Specifies the preference order for the SAProposals.
SYNTAX	unsigned 16-bit integer
VALUE	Lower-valued proposals are preferred over proposals with higher values. If two proposals have the same SequenceNumber value, then the order of preference is undefined.

7. Proposal and Transform Classes

The proposal and transform classes model the proposal settings an IPsec device will use during IKE phase 1 and 2 negotiations.



(a) ContainedTransform

7.1. The Abstract Class SAProposal

The abstract class SAProposal serves as the base class for the IKE and IPsec proposal classes. It specifies the parameters that are common to the two proposal types. The class definition for SAProposal is as follows:

NAME	SAProposal
DESCRIPTION	Specifies the common proposal parameters for IKE and IPsec security association negotiation.
ABSTRACT	TRUE
PROPERTIES	Name
	MaxLifetimeSeconds
	MaxLifetimeKilobytes

7.1.1. The Property Name

The property Name specifies a user-friendly name for the SAProposal.

The property is defined as follows:

NAME	Name
DESCRIPTION	Specifies a user-friendly name for this proposal.

```
SYNTAX       string
```

7.1.2. The Property MaxLifetimeSeconds

The property MaxLifetimeSeconds specifies the maximum amount of time, in seconds, to propose that a security association will remain valid after its creation. The property is defined as follows:

```
NAME         MaxLifetimeSeconds
DESCRIPTION  Specifies the maximum amount of time to propose a
             security association remain valid.
SYNTAX       unsigned 32-bit integer
VALUE        A value of zero indicates that the default of 8 hours
             be used.  A non-zero value indicates the maximum
             seconds lifetime.
```

7.1.3. The Property MaxLifetimeKilobytes

The property MaxLifetimeKilobytes specifies the maximum kilobyte lifetime to propose that a security association will remain valid after its creation. The property is defined as follows:

```
NAME         MaxLifetimeKilobytes
DESCRIPTION  Specifies the maximum kilobyte lifetime to propose a
             security association remain valid.
SYNTAX       unsigned 32-bit integer
VALUE        A value of zero indicates that there should be no
             maximum kilobyte lifetime.  A non-zero value specifies
             the desired kilobyte lifetime.
```

7.2. The Class IKEProposal

The class IKEProposal specifies the proposal parameters necessary to drive an IKE security association negotiation. The class definition for IKEProposal is as follows:

```
NAME         IKEProposal
DESCRIPTION  Specifies the proposal parameters for IKE security
             association negotiation.
DERIVED FROM SAProposal
ABSTRACT     FALSE
PROPERTIES   LifetimeDerivedKeys
             CipherAlgorithm
             HashAlgorithm
             PRFAlgorithm
             GroupId
             AuthenticationMethod
```

7.2.1. The Property LifetimeDerivedKeys

The property LifetimeDerivedKeys specifies the number of times that a phase 1 key will be used to derive a phase 2 key before the phase 1 security association needs renegotiated. Even though this is not

a parameter that is sent in an IKE proposal, it is included in the proposal as the number of keys derived may be a result of the strength of the algorithms in the IKE propsoal. The property is defined as follows:

NAME	LifetimeDerivedKeys
DESCRIPTION	Specifies the number of phase 2 keys that can be derived from the phase 1 key.
SYNTAX	unsigned 32-bit integer
VALUE	A value of zero indicates that there is no limit to the number of phase 2 keys which may be derived from the phase 1 key; instead the seconds and/or kilobytes lifetime will dictate the phase 1 rekeying. A non-zero value specifies the number of phase 2 keys that can be derived from the phase 1 key.

7.2.2. The Property CipherAlgorithm

The property CipherAlgorithm specifies the proposed phase 1 security association encryption algorithm. The property is defined as follows:

NAME	CipherAlgorithm
DESCRIPTION	Specifies the proposed encryption algorithm for the phase 1 security association.
SYNTAX	unsigned 16-bit integer
VALUE	1 - DES-CBC 2 - IDEA-CBC 3 - Blowfish-CBC 4 - RC5-R16-B64-CBC 5 - 3DES-CBC 6 - CAST-CBC

7.2.3. The Property HashAlgorithm

The property HashAlgorithm specifies the proposed phase 1 security assocation hash algorithm. The property is defined as follows:

NAME	HashAlgorithm
DESCRIPTION	Specifies the proposed hash algorithm for the phase 1 security association.
SYNTAX	unsigned 16-bit integer
VALUE	1 - MD5 2 - SHA-1 3 - Tiger

7.2.4. The Property PRFAlgorithm

The property PRFAlgorithm specifies the proposed phase 1 security

association psuedo-random function. The property is defined as follows:

NAME PRFAlgorithm

```
DESCRIPTION  Specifies the proposed psuedo-random function for the
             phase 1 security association.
SYNTAX       unsigned 16-bit integer
VALUE        Currently none defined.
```

7.2.5. The Property GroupId

The property GroupId specifies the proposed phase 1 security assocation Diffie-Hellman group. The property is defined as follows:

```
NAME         GroupId
DESCRIPTION  Specifies the proposed Diffie-Hellman group for the
             phase 1 security association.
SYNTAX       unsigned 16-bit integer
VALUE        1 - 768-bit MODP group
             2 - 1024-bit MODP group
             3 - EC2N group on GP[2^155]
             4 - EC2N group on GP[2^185]
             5 - 1536-bit MODP group
```

7.2.6. The Property AuthenticationMethod

The property AuthenticationMethod specifies the proposed phase 1 authentication method. The property is defined as follows:

```
NAME         AuthenticationMethod
DESCRIPTION  Specifies the proposed authentication method for the
             phase 1 security association.
SYNTAX       unsigned 16-bit integer
VALUE        0 - a special value which indicates that this
             particular proposal should be repeated once for each
             authentication method that corresponds to the
             credentials installed on the machine. For example, if
             the system has a pre-shared key and a certificate, a
             proposal list could be constructed which includes a
             proposal that specifies pre-shared key and proposals
             for any of the public-key authentication methods.
             1 - Pre-shared key
             2 - DSS signatures
             3 - RSA signatures
             4 - Encryption with RSA
             5 - Revised encryption with RSA
             6 - Kerberos (has this number been assigned???)
```

7.3. The Class IPsecProposal

The class IPsecProposal adds no new properties, but inherits proposal propoerties from SAProposal as well as aggregating the

security association transforms necessary for building an IPsec proposal (see the aggregation class ContainedTransform). The class definition for IPsecProposal is as follows:

```
NAME         IPsecProposal
DESCRIPTION  Specifies the proposal parameters for IPsec security
             association negotiation.
DERIVED FROM SAProposal
ABSTRACT     FALSE
```

7.4. The Abstract Class SATransform

The abstract class SATransform serves as the base class for the IPsec transforms that can be used to compose an IPsec proposal. The class definition for SATransform is as follows:

```
NAME         SATransform
DESCRIPTION  Base class for the different IPsec transforms.
ABSTRACT     TRUE
PROPERTIES   Name
             VendorID
```

7.4.1. The Property Name

The property Name specifies a user-friendly name for the SATransform. The property is defined as follows:

```
NAME         Name
DESCRIPTION  Specifies a user-friendly name for this transform.
SYNTAX       string
```

7.4.1. The Property VendorID

The property VendorID specifies the vendor ID for vendor-defined transforms. The property is defined as follows:

```
NAME         VendorID
DESCRIPTION  Specifies the vendor ID for vendor-defined transforms.
SYNTAX       string
VALUE        An empty VendorID string indicates that the transform
             is one of the previously-defined ones.
```

7.5. The Class AHTransform

The class AHTransform specifies the AH algorithm to propose during IPsec security association negotiation. The class definition for AHTransform is as follows:

```
NAME         AHTransform
DESCRIPTION  Specifies the AH algorithm to propose.
ABSTRACT     FALSE
PROPERTIES   AHTransformId
```

7.5.1. The Property AHTransformId

The property AHTransformId specifies the transform ID of the AH algorithm to propose. The property is defined as follows:

```
NAME         AHTransformId
DESCRIPTION  Specifies the transform ID of the AH algorithm.
SYNTAX       unsigned 16-bit integer
VALUE        2 - MD5
             3 - SHA-1
             4 - DES
```

7.6. The Class ESPTransform

The class ESPTransform specifies the ESP algorithms to propose during IPsec security association negotiation. The class definition for ESPTransform is as follows:

```
NAME         ESPTransform
DESCRIPTION  Specifies the ESP algorithms to propose.
ABSTRACT     FALSE
PROPERTIES   IntegrityTransformId
             CipherTransformId
             CipherKeyLength
             CipherKeyRounds
```

7.6.1. The Property IntegrityTransformId

The property IntegrityTransformId specifies the transform ID of the ESP integrity algorithm to propose. The property is defined as follows:

```
NAME         IntegrityTransformId
DESCRIPTION  Specifies the transform ID of the ESP integrity
             algorithm.
SYNTAX       unsigned 16-bit integer
VALUE        0 - None
             1 - HMAC-MD5
             2 - HMAC-SHA
             3 - DES-MAC
             4 - KPDK
```

7.6.2. The Property CipherTransformId

The property CipherTransformId specifies the transform ID of the ESP encryption algorithm to propose. The property is defined as follows:

```
NAME         CipherTransformId
DESCRIPTION  Specifies the transform ID of the ESP encryption
             algorithm.
SYNTAX       unsigned 16-bit integer
VALUE        1 - DES IV64
```

2 - DES
3 - 3DES
4 - RC5
5 - IDEA

```
6 - CAST
7 - Blowfish
8 - 3IDEA
9 - DES IV32
10 - RC4
11 - NULL
```

7.6.3. The Property CipherKeyLength

The property CipherKeyLength specifies, in bits, the key length for the ESP encryption algorithm. For encryption algorithms which use fixed-length keys, this value is ignored. The property is defined as follows:

NAME	CipherKeyLength
DESCRIPTION	Specifies the ESP encryption key length in bits.
SYNTAX	unsigned 16-bit integer

7.6.4. The Property CipherKeyRounds

The property CipherKeyRounds specifies the number of key rounds for the ESP encryption algorithm. The property is defined as follows:

NAME	CipherKeyRounds
DESCRIPTION	Specifies the number of key rounds for the ESP encryption algorithm.
SYNTAX	unsigned 16-bit integer
VALUE	Currently, key rounds are not defined for any ESP encryption algorithms.

7.7. The Class IPCOMPTransform

The class IPCOMPTransform specifies the IP compression (IPCOMP) algorithm to propose during IPsec security association negotiation. The class definition for IPCOMPTransform is as follows:

NAME	IPCOMPTransform
DESCRIPTION	Specifies the IPCOMP algorithm to propose.
ABSTRACT	FALSE
PROPERTIES	Algorithm DictionarySize PrivateAlgorithm

7.7.1. The Property Algorithm

The property Algorithm specifies the transform ID of the IPCOMP compression algorithm to propose. The property is defined as follows:

```
   NAME         Algorithm
   DESCRIPTION  Specifies the transform ID of the IPCOMP compression
                algorithm.
   SYNTAX       unsigned 16-bit integer
```

```
VALUE       1 - OUI (the property PrivateAlgorithm will contain the
            vendor-specific algorithm to use)
            2 - DEFLATE
            3 - LZS
            4 - V42BIS (has this number been assigned ???)
```

7.7.2. The Property DictionarySize

The property DictionarySize specifies the log2 maximum size of the diction for the compression algorithm. For compression algorithms that have pre-defined dictionary sizes, this value is ignores. The property is defined as follows:

```
NAME         DictionarySize
DESCRIPTION  Specifies the log2 maximum size of the dictionary.
SYNTAX       unsigned 16-bit integer
```

7.7.3. The Property PrivateAlgorithm

The property PrivateAlgorithm specifies a private vendor-specific compression algorithm. This value is only used when the property Algorithm is 1 (OUI). The property is defined as follows:

```
NAME         PrivateAlgorithm
DESCRIPTION  Specifies a private vendor-specific compression
             algorithm.
SYNTAX       unsigned 32-bit integer
```

7.8. The Aggregation Class ContainedTransform

The class ContainedTransform associates an IPsecProposal with the set of SATransforms that make up the proposal. If multiple tranforms of the same type are in a proposal, then they are to be logically ORed and the order of preference is dictated by the SequenceNumber property. Sets of transforms of different types are logically ANDed. For example, if the proposal list were

```
ESP = { (HMAC-MD5, DES), (HMAC-MD5, 3DES) }
AH  = { MD5, SHA-1 }
```

then the one sending the proposal wants the other side to pick one from the ESP transform list AND one from the AH transform list. The class definition for ContainedProposal is as follows:

```
NAME         ContainedTransform
DESCRIPTION  Associates an IPsecProposal with the set of
             SATransforms that make up the proposal.
ABSTRACT     FALSE
```

```
   PROPERTIES   GroupComponent[ref IPsecProposal[0..n]]
                PartComponent[ref SATransform[1..n]]
                SequenceNumber
```

7.8.1. The Reference GroupComponent

The property GroupComponent contains an object reference to an IPsecProposal that contains one or more SATransforms. The [0..n] cardinality indicates that there may be zero or more IPsecProposals that contain any given SATransform.

7.8.2. The Reference PartComponent

The property PartComponent contains an object reference to an SATransform contained by one or more IPsecProposals. The [1..n] cardinality indicates that an IPsecPropsal MUST contain at least one SATransform.

7.8.3. The Property SequenceNumber

The property SequenceNumber specifies the order of preference for the SATransforms of the same type. The property is defined as follows:

NAME	SequenceNumber
DESCRIPTION	Specifies the preference order for the SATransforms of the same type.
SYNTAX	unsigned 16-bit integer
VALUE	Lower-valued transforms are preferred over transforms of the same type with higher values. If two transforms of the same type have the same SequenceNumber value, then the order of preference is undefined.

8. Security Considerations

This document describes a schema for IPsec policy. It does not detail security requirements for storage or delivery of said schema. Storage and delivery security requirements should be detailed in a comprehensive security policy architecture document.

9. 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.

10. Acknowledgments

The author would like to thank Mike Jeronimo, Ylian Saint-Hilaire, Vic Lortz, and William Dixon for their contributions to this IPsec policy model.

Additionally, this draft would not have been possible without the preceding IPsec schema drafts. For that, thanks go out to Rob Adams, Partha Bhattacharya, William Dixon, Roy Pereira, and Raju Rajan.

11. References

[IKE] Harkins, D., and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[COMP] Shacham, A., and R. Monsour, R. Pereira, M. Thomas, "IP Payload Compression Protocol (IPComp)", RFC 2393, August 1998.

[ESP] Kent, S., and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[AH] Kent, S., and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[PCIM] Moore, B., and E. Ellesson, J. Strassner, "Policy Core Information Model -- Version 1 Specification", draft-ietf-policy-core-infor-model-06.txt, May 2000. Internet-Draft work in progress.

[DOI] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

[LDAP] Wahl, M., and T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[COPS] Boyle, J., and R. Cohen, D. Durham, S. Herzog, R. Rajan, A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000. Internet-Draft work in progress.

[COPSPR] Chan, K., and D. Durham, S. Gai, S. Herzog, K. McCloghrie, F. Reichmeyer, J. Seligson, A. Smith, R. Yavatkar, "COPS Usage for Policy Provisioning", draft-ietf-rap-pr-02.txt, March 2000. Internet-Draft work in progress.

[SPSL] Condell, M., and C. Lynn, J. Zao, "Security Policy Specification Language", draft-ietf-ipsp-spsl-00.txt, March 2000. Internet-Draft work in progress.

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

12. Disclaimer

The views and specification herein are those of the authors and are not necessarily those of their employer. The authors and their employer specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this specification.

13. Author's Address

Jamie Jason
Intel Corporation
MS JF3-206
2111 NE 25th Ave.
Hillsboro, OR 97124
Phone: +1-503-264-9531
Fax: +1-503-264-9428
E-Mail: jamie.jason@intel.com

14. Full Copyright Statement

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.