

Internet Engineering Task Force
INTERNET DRAFT
August-2002

Jamie Jason
Intel Corporation
Lee Rafalow
IBM
Eric Vyncke
Cisco Systems

IPsec Configuration Policy Model
[draft-ietf-ipsec-config-policy-model-06.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 information model of IPsec policy designed to:

- o facilitate agreement about the content and semantics of IPsec policy
- o 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 information model described in this document models the configuration parameters defined by the IP Security protocol [COMP, ESP, AH]. The information model also covers the parameters found by the Internet Key Exchange [DOI, IKE] protocol. Other key exchange protocols could be easily added to the information model by a simple extension. Other extensions can further be added easily due to the object-oriented nature of the model.

This information model is based upon the core policy classes as defined in the Policy Core Information Model (PCIM) [[PCIM](#)] and on the Policy Core Information Model Extensions (PCIME) [[PCIME](#)].

Table of Contents

Status of this Memo.....	1
Abstract.....	1
Table of Contents.....	2
1 . Introduction.....	10
1 . Introduction.....	10
2 . UML Conventions.....	10
3 . IPsec Policy Model Inheritance Hierarchy.....	11
4 . Policy Classes.....	16
4.1 . The Class IPsecPolicyGroup.....	17
4.2 . The Class SARule.....	18
4.2.1 . The Properties PolicyRuleName, Enabled, ConditionListType, RuleUsage, Mandatory, SequencedActions, PolicyRoles, and PolicyDecisionStrategy.....	18
4.2.2 The Property ExecutionStrategy.....	18
4.2.3 The Property LimitNegotiation.....	20
4.3 . The Class IKERule.....	21
4.3.1 . The Property IdentityContexts.....	21
4.4 . The Class IPsecRule.....	22
4.5 . The Association Class IPsecPolicyForEndpoint.....	22
4.5.1 . The Reference Antecedent.....	22
4.5.2 . The Reference Dependent.....	22
4.6 . The Association Class IPsecPolicyForSystem.....	22
4.6.1 . The Reference Antecedent.....	23
4.6.2 . The Reference Dependent.....	23
4.7 . The Aggregation Class SARuleInPolicyGroup.....	23
4.7.1 . The Property Priority.....	23
4.7.2 . The Reference GroupComponent.....	23
4.7.3 . The Reference PartComponent.....	23
4.8 . The Aggregation Class SAConditionInRule.....	24
4.8.1 . The Properties GroupNumber and ConditionNegated.....	24
4.8.2 . The Reference GroupComponent.....	24
4.8.3 . The Reference PartComponent.....	25
4.9 . The Aggregation Class PolicyActionInSARule.....	25
4.9.1 . The Reference GroupComponent.....	25
4.9.2 . The Reference PartComponent.....	25
4.9.3 . The Property ActionOrder.....	25
5 . Condition and Filter Classes.....	26
5.1 . The Class SACondition.....	26
5.2 . The Class IPHeadersFilter.....	27
5.3 . The Class CredentialFilterEntry.....	27
5.3.1 . The Property MatchFieldName.....	27
5.3.2 . The Property MatchFieldValue.....	28
5.3.3 . The Property CredentialType.....	28
5.4 . The Class IPSOFilterEntry.....	28
5.4.1 . The Property MatchConditionType.....	29
5.4.2 . The Property MatchConditionValue.....	29
5.5 . The Class PeerIDPayloadFilterEntry.....	29

5.5.1. The Property MatchIdentityType.....	30
5.5.2. The Property MatchIdentityValue.....	30
5.6. The Association Class FilterOfSACondition.....	31
5.6.1. The Reference Antecedent.....	31
5.6.2. The Reference Dependent.....	31
5.7. The Association Class AcceptCredentialFrom.....	31
5.7.1. The Reference Antecedent.....	32
5.7.2. The Reference Dependent.....	32
6. Action Classes.....	33
6.1. The Class SAAction.....	34
6.1.1. The Property DoActionLogging.....	34
6.1.2. The Property DoPacketLogging.....	34
6.2. The Class SASStaticAction.....	35

6.2.1. The Property LifetimeSeconds.....	35
6.3. The Class IPsecBypassAction.....	35
6.4. The Class IPsecDiscardAction.....	35
6.5. The Class IKERejectAction.....	36
6.6. The Class PreconfiguredSAAction.....	36
6.6.1. The Property LifetimeKilobytes.....	36
6.7. The Class PreconfiguredTransportAction.....	37
6.8. The Class PreconfiguredTunnelAction.....	37
6.8.1. The Property DFHandling.....	37
6.9. The Class SANegotiationAction.....	37
6.10. The Class IKENegotiationAction.....	38
6.10.1. The Property MinLifetimeSeconds.....	38
6.10.2. The Property MinLifetimeKilobytes.....	38
6.10.3. The Property IdleDurationSeconds.....	39
6.11. The Class IPsecAction.....	40
6.11.1. The Property UsePFS.....	40
6.11.2. The Property UseIKEGroup.....	40
6.11.3. The Property GroupId.....	40
6.11.4. The Property Granularity.....	41
6.11.5. The Property VendorID.....	41
6.12. The Class IPsecTransportAction.....	41
6.13. The Class IPsecTunnelAction.....	41
6.13.1. The Property DFHandling.....	42
6.14. The Class IKEAction.....	42
6.14.1. The Property ExchangeMode.....	42
6.14.2. The Property UseIKEIdentityType.....	43
6.14.3. The Property VendorID.....	43
6.14.4. The Property AggressiveModeGroupId.....	43
6.15. The Class PeerGateway.....	43
6.15.1. The Property Name.....	44
6.15.2. The Property PeerIdentityType.....	44
6.15.3. The Property PeerIdentity.....	44
6.16. The Association Class PeerGatewayForTunnel.....	44
6.16.1. The Reference Antecedent.....	45
6.16.2. The Reference Dependent.....	45
6.16.3. The Property SequenceNumber.....	45
6.17. The Aggregation Class ContainedProposal.....	45
6.17.1. The Reference GroupComponent.....	46
6.17.2. The Reference PartComponent.....	46
6.17.3. The Property SequenceNumber.....	46
6.18. The Association Class HostedPeerGatewayInformation.....	46
6.18.1. The Reference Antecedent.....	46
6.18.2. The Reference Dependent.....	46
6.19. The Association Class TransformOfPreconfiguredAction.....	46
6.19.1. The Reference Antecedent.....	47
6.19.2. The Reference Dependent.....	47
6.19.3. The Property SPI.....	47
6.19.4. The Property Direction.....	47
6.20 The Association Class PeerGatewayForPreconfiguredTunnel.....	47

6.20.1.	The Reference Antecedent.....	48
6.20.2.	The Reference Dependent.....	48
7.	Proposal and Transform Classes.....	49
7.1.	The Abstract Class SAProposal.....	49
7.1.1.	The Property Name.....	49
7.2.	The Class IKEProposal.....	49
7.2.1.	The Property CipherAlgorithm.....	50
7.2.2.	The Property HashAlgorithm.....	50
7.2.3.	The Property PRFAlgorithm.....	50
7.2.4.	The Property GroupId.....	51
7.2.5.	The Property AuthenticationMethod.....	51
7.2.6.	The Property MaxLifetimeSeconds.....	51
7.2.7.	The Property MaxLifetimeKilobytes.....	52

7.2.8.	The Property VendorID.....	52
7.3.	The Class IPsecProposal.....	52
7.4.	The Abstract Class SATransform.....	52
7.4.1.	The Property CommonName.....	52
7.4.2.	The Property VendorID.....	53
7.4.3.	The Property MaxLifetimeSeconds.....	53
7.4.4.	The Property MaxLifetimeKilobytes.....	53
7.5.	The Class AHTransform.....	53
7.5.1.	The Property AHTransformId.....	54
7.5.2.	The Property UseReplayPrevention.....	54
7.5.3.	The Property ReplayPreventionWindowSize.....	54
7.6.	The Class ESPTransform.....	54
7.6.1.	The Property IntegrityTransformId.....	54
7.6.2.	The Property CipherTransformId.....	55
7.6.3.	The Property CipherKeyLength.....	55
7.6.4.	The Property CipherKeyRounds.....	55
7.6.5.	The Property UseReplayPrevention.....	55
7.6.6.	The Property ReplayPreventionWindowSize.....	55
7.7.	The Class IPCOMPTransform.....	56
7.7.1.	The Property Algorithm.....	56
7.7.2.	The Property DictionarySize.....	56
7.7.3.	The Property PrivateAlgorithm.....	56
7.8.	The Association Class SAProposalInSystem.....	56
7.8.1.	The Reference Antecedent.....	57
7.8.2.	The Reference Dependent.....	57
7.9.	The Aggregation Class ContainedTransform.....	57
7.9.1.	The Reference GroupComponent.....	57
7.9.2.	The Reference PartComponent.....	57
7.9.3.	The Property SequenceNumber.....	57
7.10.	The Association Class SATransformInSystem.....	58
7.10.1.	The Reference Antecedent.....	58
7.10.2.	The Reference Dependent.....	58
8.	IKE Service and Identity Classes.....	59
8.1.	The Class IKEService.....	60
8.2.	The Class PeerIdentityTable.....	60
8.2.1.	The Property Name.....	60
8.3.	The Class PeerIdentityEntry.....	60
8.3.1.	The Property PeerIdentity.....	61
8.3.2.	The Property PeerIdentityType.....	61
8.3.3.	The Property PeerAddress.....	61
8.3.4.	The Property PeerAddressType.....	61
8.4.	The Class AutostartIKEConfiguration.....	61
8.5.	The Class AutostartIKESetting.....	62
8.5.1.	The Property Phase1Only.....	62
8.5.2.	The Property AddressType.....	62
8.5.3.	The Property SourceAddress.....	63
8.5.4.	The Property SourcePort.....	63
8.5.5.	The Property DestinationAddress.....	63
8.5.6.	The Property DestinationPort.....	63

8.5.7. The Property Protocol.....	63
8.6. The Class IKEIdentity.....	63
8.6.1. The Property IdentityType.....	64
8.6.2. The Property IdentityValue.....	64
8.6.3. The Property IdentityContexts.....	64
8.7. The Association Class HostedPeerIdentityTable.....	65
8.7.1. The Reference Antecedent.....	65
8.7.2. The Reference Dependent.....	65
8.8. The Aggregation Class PeerIdentityMember.....	65
8.8.1. The Reference Collection.....	66
8.8.2. The Reference Member.....	66
8.9. The Association Class IKEServicePeerGateway.....	66
8.9.1. The Reference Antecedent.....	66

8.9.2. The Reference Dependent.....	66
8.10. The Association Class IKEServicePeerIdentityTable.....	66
8.10.1. The Reference Antecedent.....	67
8.10.2. The Reference Dependent.....	67
8.11. The Association Class IKEAutostartSetting.....	67
8.11.1. The Reference Element.....	67
8.11.2. The Reference Setting.....	67
8.12. The Aggregation Class AutostartIKESettingContext.....	67
8.12.1. The Reference Context.....	67
8.12.2. The Reference Setting.....	68
8.12.3. The Property SequenceNumber.....	68
8.13. The Association Class IKEServiceForEndpoint.....	68
8.13.1. The Reference Antecedent.....	68
8.13.2. The Reference Dependent.....	68
8.14. The Association Class IKEAutostartConfiguration.....	68
8.14.1. The Reference Antecedent.....	69
8.14.2. The Reference Dependent.....	69
8.14.3. The Property Active.....	69
8.15. The Association Class IKEUsesCredentialManagementService....	69
8.15.1. The Reference Antecedent.....	70
8.15.2. The Reference Dependent.....	70
8.16. The Association Class EndpointHasLocalIKEIdentity.....	70
8.16.1. The Reference Antecedent.....	70
8.16.2. The Reference Dependent.....	70
8.17. The Association Class CollectionHasLocalIKEIdentity.....	70
8.17.1. The Reference Antecedent.....	71
8.17.2. The Reference Dependent.....	71
8.18. The Association Class IKEIdentityCredential.....	71
8.18.1. The Reference Antecedent.....	71
8.18.2. The Reference Dependent.....	71
9. Implementation Requirements.....	71
10. Security Considerations.....	75
11. Intellectual Property.....	75
12. Acknowledgments.....	76
13. References.....	76
14. Disclaimer.....	77
15. Authors' Addresses.....	77
16. Full Copyright Statement.....	77

1. Introduction

IP 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:

- o a Lightweight Directory Access Protocol (LDAP) [[LDAP](#)] schema in a directory
- o an on-the-wire representation over a transport protocol like the Common Object Policy Service (COPS) [[COPS](#), [COPSPR](#)]
- o a text-based policy specification language suitable for editing by an administrator
- o 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. This document captures this concept and introduces a task-independent canonical representation for IPsec policies.

In order to have a simple information model, this document focuses mainly on the existing protocols [[COMP](#), [ESP](#), [AH](#), [DOI](#), [IKE](#)]. The model can easily be extended if needed due to its object-oriented nature.

This document is organized as follows:

- o [Section 2](#) provides a quick introduction to the Unified Modeling Language (UML) graphical notation conventions used in this document.
- o [Section 3](#) provides the inheritance hierarchy that describes where the IPsec policy classes fit into the policy class hierarchy already defined by the Policy Core Information Model (PCIM) and Policy Core Information Model Extensions (PCIMe).
- o Sections [4](#) through [8](#) describes the class that make up the IPsec policy model.
- o [Section 9](#) presents the implementation requirements for the classes in the model (i.e., the MUST/MAY/SHOULD status).

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:

- o Boxes represent classes, with class names in brackets ([]) representing an abstract 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 to model a relationship between two classes. Classes that share an association are connected using a line. A special kind of association is also used: an aggregation. An aggregation models a whole-part relationship between two classes. Associations, and therefore aggregations, can also be modeled as classes.
- o A line that begins with an "o" denotes aggregation. Aggregation denotes containment in which the contained class and the containing class have independent lifetimes.
- o Next to a line representing an association appears a cardinality. Cardinalities indicate the constraints on the number of object instances in a set of relationships. Every association instance has a single set of references. The cardinality indicates the number of instances that may refer to a given object instance. The cardinality 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.
- o A class that has an association may have a "w" next to the line representing the association. This is called a weak association and is discussed in [[PCIM](#)].

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 Hierarchy

Like PCIM and PCIME from which it is derived, the IPsec Configuration Policy Model derives from and uses classes defined in the DMTF [[DMTF](#)] Common Information Model (CIM). The following tree represents the inheritance hierarchy for the IPsec policy model classes and how they fit into PCIM, PCIME and the other DMTF models (see Appendices for descriptions of classes that are not being introduced as part of IPsec model). CIM classes that are not used as a superclass from which to derive new classes but are only referenced are not included

this inheritance hierarchy, but can be found in the appropriate DMTF document [[CIMCORE](#)], [[CIMUSER](#)] or [CIMNETWORK].

```
ManagedElement (DMTF Core Model - [CIMCORE])
|
+--Collection (DMTF Core Model - [CIMCORE])
|  |
|  +--PeerIdentityTable
|
+--ManagedSystemElement (DMTF Core Model - [CIMCORE])
|  |
|  +--LogicalElement (DMTF Core Model - [CIMCORE])
|      |
```

```
|      +--FilterEntryBase (DMTF Network Model - [CIMNETWORK])
|      |
|      |      +--CredentialFilterEntry
|      |      |
|      |      +--IPHeadersFilter (PCIMe)
|      |      |
|      |      +--IPSOFilterEntry
|      |      |
|      |      +--PeerIDPayloadFilterEntry
|      |
|      +--PeerGateway
|      |
|      +--PeerIdentityEntry
|      |
|      +--Service (DMTF Core Model - [CIMCORE])
|      |
|      +--IKEService
|
+--OrganizationalEntity (DMTF User Model - [CIMUSER])
|
|      +--UserEntity (DMTF User Model - [CIMUSER])
|      |
|      |      +--UsersAccess (DMTF User Model - [CIMUSER])
|      |      |
|      |      +--IKEIdentity
|      |
|      +--Policy (PCIM)
|      |
|      |      +--PolicyAction (PCIM)
|      |      |
|      |      |      +--CompoundPolicyAction (PCIMe)
|      |      |      |
|      |      |      +--SAAction
|      |      |      |
|      |      |      |      +--SANegotiationAction
|      |      |      |      |
|      |      |      |      |      +--IKENegotiationAction
|      |      |      |      |      |
|      |      |      |      |      +--IKEAction
|      |      |      |      |      |
|      |      |      |      |      +--IPsecAction
|      |      |      |      |      |
|      |      |      |      |      +--IPsecTransportAction
|      |      |      |      |      |
|      |      |      |      |      +--IPsecTunnelAction
|      |      |      |      |
|      |      |      |      +--SASStaticAction
|      |      |      |
|      |      |      +--IKERejectAction
```

```

| | |
| | |---IPsecBypassAction
| | |
| | |---IPsecDiscardAction
| | |
| | |---PreconfiguredSAAction
| | |
| | |---PreconfiguredTransportAction
| | |
| | |---PreconfiguredTunnelAction
| | |
| | |---PolicyCondition (PCIM)
| | |

```



```

| | +--SACondition
| |
| +--PolicySet (PCIMe)
| | |
| | +--PolicyGroup (PCIM & PCIME)
| | | |
| | | +--IPsecPolicyGroup
| | | |
| | +--PolicyRule (PCIM & PCIME)
| | |
| | | +--SARule
| | | |
| | | | +--IKERule
| | | | |
| | | | +--IPsecRule
| | |
| +--SAProposal
| | |
| | +--IKEProposal
| | |
| | +--IPsecProposal
| | |
| +--SATransform
| | |
| | +--AHTransform
| | |
| | +--ESPTransform
| | |
| | +--IPCOMPTransform
|
+--Setting (DMTF Core Model - [CIMCORE])
| |
| +--SystemSetting (DMTF Core Model - [CIMCORE])
| | |
| | +--AutostartIKESetting
|
+--SystemConfiguration (DMTF Core Model - [CIMCORE])
|
+--AutostartIKEConfiguration

```

The following tree represents the inheritance hierarchy of the IPsec policy model association classes and how they fit into PCIM and the other DMTF models (see Appendices for description of associations classes that are not being introduced as part of IPsec model).

```

Dependency (DMTF Core Model - [CIMCORE])
|
+--AcceptCredentialsFrom
|

```

```
+--ElementAsUser (DMTF User Model - [CIMUSER])
| |
| +--EndpointHasLocalIKEIdentity
| |
| +--CollectionHasLocalIKEIdentity
|
+--FilterOfSACondition
|
+--HostedPeerGatewayInformation
|
+--HostedPeerIdentityTable
|
+--IKEAutostartConfiguration
```

```
|
+--IKEServiceForEndpoint
|
+--IKEServicePeerGateway
|
+--IKEServicePeerIdentityTable
|
+--IKEUsesCredentialManagementService
|
+--IPsecPolicyForEndpoint
|
+--IPsecPolicyForSystem
|
+--PeerGatewayForPreconfiguredTunnel
|
+--PeerGatewayForTunnel
|
+--PolicyInSystem (PCIM)
|  |
|  +--SAProposalInSystem
|  |
|  +--SATransformInSystem
|
+--TransformOfPreconfiguredAction
|
+--UsersCredential (DMTF User Model - [CIMUSER])
|
|  +--IKEIdentityCredential

ElementSetting (DMTF Core Model - [CIMCORE])
|
+--IKEAutostartSetting

MemberOfCollection (DMTF Core Model - [CIMCORE])
|
+--PeerIdentityMember

PolicyComponent (PCIM)
|
+--ContainedProposal
|
+--ContainedTransform
|
+--PolicyActionStructure (PCIMe)
|  |
|  +--PolicyActionInPolicyRule (PCIM & PCIMe)
|  |
|  +--PolicyActionInSARule
|
```

```

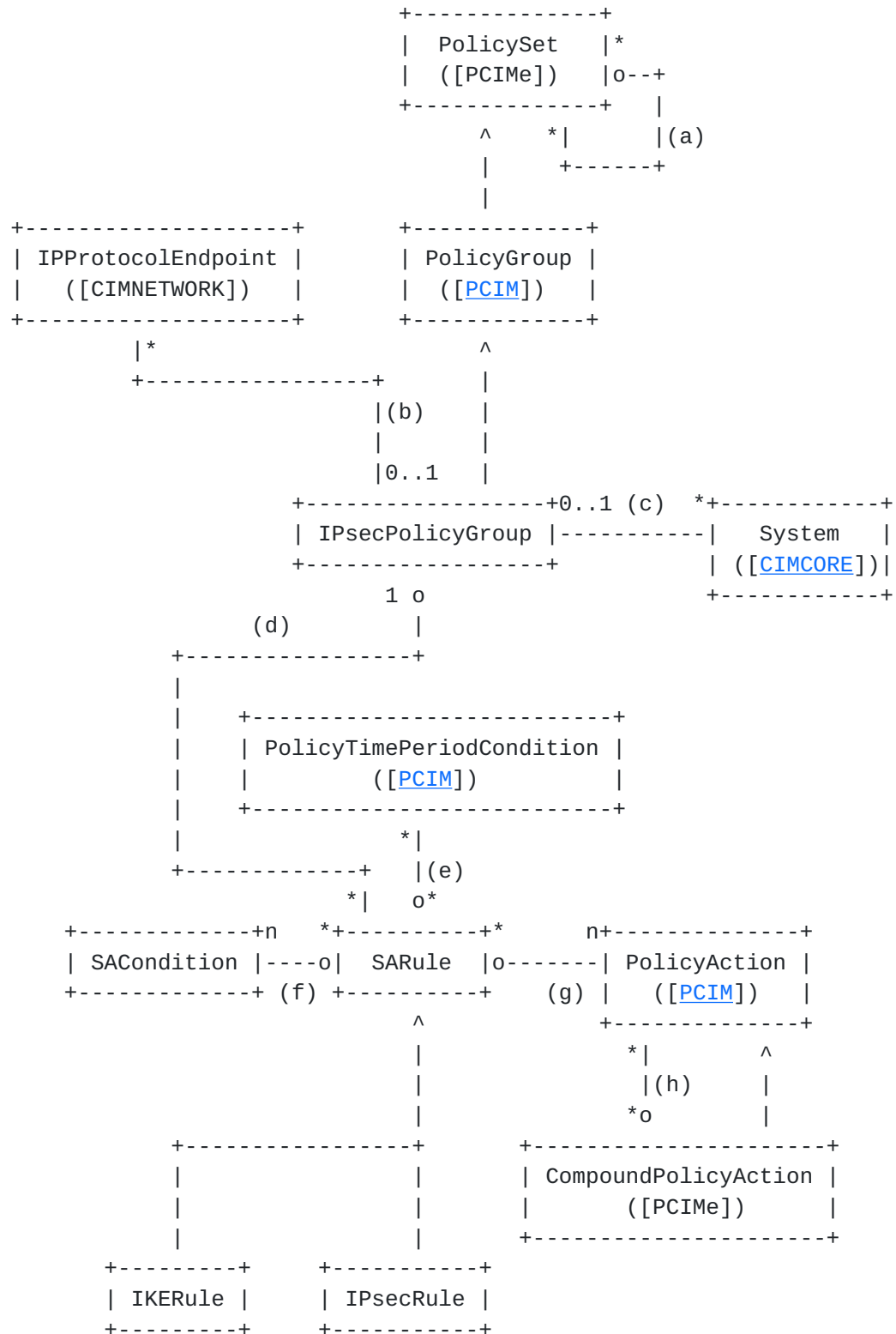
+--PolicyConditionStructure (PCIMe)
| |
| +--PolicyConditionInPolicyRule (PCIM & PCIMe)
|   |
|   +--SAConditionInRule
|
+--PolicySetComponent (PCIMe)
|
| +--SARuleInPolicyGroup

SystemSettingContext (DMTF Core Model - [CIMCORE])
|
+--AutostartIKESettingContext

```

4. Policy Classes

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



- (a) PolicySetComponent ([PCIMe])
- (b) IPsecPolicyForEndpoint
- (c) IPsecPolicyForSystem
- (d) SARuleInPolicyGroup
- (e) PolicyRuleValidityPeriod ([[PCIM](#)])
- (f) SAConditionInRule
- (g) PolicyActionInSARule
- (h) PolicyActionInPolicyAction ([PCIMe])

An IPsecPolicyGroup represents the set of policies that are used on an interface. This IPsecPolicyGroup SHOULD be associated either directly with the IPProtocolEndpoint class instance that represents

the interface (via the `IPsecPolicyForEndpoint` association) or indirectly (via the `IPsecPolicyForSystem` association) associated with the System that hosts the interface.

The IKE and IPsec rules are used to build or to negotiate the IPsec SADB. The IPsec rules represent the Security Policy Database. The SADB itself is not modeled by this document.

The IKE and IPsec rules usage can be described as (see also [section 6](#) about actions):

- o an egress unprotected packet will first be checked against the IPsec rules. If a match is found, the SADB will be checked. If there is no corresponding IPsec SA in the SADB and if IKE negotiation is required by the IPsec rule, the corresponding IKE rules will be used. The negotiated or preconfigured SA will then be installed in the SADB.
- o An ingress unprotected packet will first be checked against the IPsec rules. If a match is found, the SADB will be checked for a corresponding IPsec SA. If there is no corresponding IPsec SA and a preconfigured SA exists, this preconfigured SA will be installed in the IPsec SADB. This behavior should only apply to bypass and discard actions.
- o An ingress protected packet will first be checked against the IPsec rules. If a match is found, the SADB will be checked for a corresponding IPsec SA. If there is no corresponding IPsec SA and a preconfigured SA exists, this preconfigured SA will be installed in the IPsec SADB.
- o An ingress IKE negotiation packet, which is not part of an existing IKE SA, will be checked against the IKE rules. The `SACondition` for the `IKERule` will usually be composed of a `PeerIDPayloadFilterEntry` (typically for a aggressive mode IKE negotiation) or a `IPHeadersFilter`. The negotiated SA will then be installed in the SADB.

It is expected that when a IKE negotiation has to be initiated when required by an IPsec rule, the set of IKE rules will be checked. The IKE rules check will be based on the outgoing IKE packet using `IPHeadersFilter` entries (typically using the `HdrDstAddress` property).

[4.1.1. The Class `IPsecPolicyGroup`](#)

The class `IPsecPolicyGroup` serves as a container of either other `IPsecPolicyGroups` or a set of `SARules`. The class definition for `IPsecPolicyGroup` is as follows:

NAME	<code>IPsecPolicyGroup</code>
DESCRIPTION	Either a set of <code>IPsecPolicyGroups</code> or a set of <code>SARules</code> .
DERIVED FROM	<code>PolicyGroup</code> (see [PCIM] & [PCIMe])

ABSTRACT FALSE
PROPERTIES PolicyGroupName (from PolicyGroup)
PolicyDecisionStrategy (from PolicySet)
PolicyRoles (from PolicySet)

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 PolicySetComponent 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 PolicySetComponent aggregation for information on merging multiple IPsecPolicyGroups.

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. Through its derivation from PolicyRule, a SARule (and therefore IKERule and IPsecRule) also has the PolicyRuleValidityPeriod association.

Each valid IPsecPolicyGroup MUST contain SARules that each have a unique associated priority number in PolicySetComponent.Priority. The class definition for SARule is as follows:

NAME	SARule
DESCRIPTION	A base class for IKERule and IPsecRule.
DERIVED FROM	PolicyRule (see [PCIM] & [PCIME])
ABSTRACT	FALSE
PROPERTIES	PolicyRuleName (from PolicyRule) Enabled (from PolicyRule) ConditionListType (from PolicyRule) RuleUsage (from PolicyRule) Mandatory (from PolicyRule) SequencedActions (from PolicyRule) ExecutionStrategy (from PolicyRule) PolicyRoles (from PolicySet) PolicyDecisionStrategy (from PolicySet) LimitNegotiation

4.2.1. The Properties PolicyRuleName, Enabled, ConditionListType, RuleUsage, Mandatory, SequencedActions, PolicyRoles, and PolicyDecisionStrategy

For a description of these properties, see [[PCIM](#)] and [[PCIME](#)].

In SARule subclass instances:

- if the property Mandatory exists, it MUST be set to "true"
- if the property SequencedActions exists, it MUST be set to "mandatory"
- the property PolicyRoles is not used in the device-level model
- if the property PolicyDecisionStrategy exists, it must be set to "FirstMatching"

4.2.2 The Property ExecutionStrategy

The ExecutionStrategy properties in the PolicyRule subclasses (and in the CompoundPolicyAction class) determine the behavior of the contained actions. It defines the strategy to be used in executing the sequenced actions aggregated by a rule or a compound action. In the case of actions within a rule, the PolicyActionInSARule

aggregation is used to collect the actions into an ordered set; in the case of a compound action, the PolicyActionInPolicyAction aggregation is used to collect the actions into an ordered subset.

There are three execution strategies: do until success, do all and do until failure.

"Do Until Success" causes the execution of actions according to the ActionOrder property in the aggregation instances until a successful execution of a single action. These actions may be evaluated to determine if they are appropriate to execute rather than blindly trying each of the actions until one succeeds. For an initiator,

they are tried in the ActionOrder until the list is exhausted or one completes successfully. For example, an IKE initiator may have several IKEActions for the same SACondition. The initiator will try all IKEActions in the order defined by ActionOrder. I.e. it will possibly try several phase 1 negotiations possibly with different modes (main mode then aggressive mode) and/or with possibly multiple IKE peers. For a responder, when there is more than one action in the rule with "do until success" condition clause this provides alternative actions depending on the received proposals. For example, the same IKERule may be used to handle aggressive mode and main mode negotiations with different actions. The responder uses the first appropriate action in the list of actions.

"Do All" causes the execution all of the actions in aggregated set according to their defined order. The execution continues regardless of failures.

"Do Until Failure" causes the execution of all actions according to predefined order until the first failure in execution of an action instance. Please note that if all actions are successful then the aggregated result is a failure. This execution strategy is inherited from [\[PCIME\]](#) and is not expected to be of any use for IPsec configuration.

For example, in a nested SAs case the actions of an initiator's rule might be structured as:

```
IPsecRule.ExecutionStrategy='Do All'
|
+---1--- IPsecTunnelAction    // set up SA from host to gateway
|
+---2--- IPsecTransportAction // set up SA from host through
                        // tunnel to remote host
```

Another example, showing a rule with fallback actions might be structured as:

```
IPsecRule.ExecutionStrategy='Do Until Success'
|
+---6--- IPsecTransportAction // negotiate SA with peer
|
+---9--- IPsecBypassAction    // but if you must, allow in the clear
```

The CompoundPolicyAction class (See [\[PCIME\]](#)) may be used in constructing the actions of IKE and IPsec rules when those rules specify both multiple actions and fallback actions. The ExecutionStrategy property in CompoundPolicyAction is used in conjunction with that in the PolicyRule.

For example, in nesting SAs with a fallback security gateway, the actions of a rule might be structured as:

```
IPsecRule.ExecutionStrategy='Do All'
|
+---1--- CompoundPolicyAction.ExecutionStrategy='Do Until Success'
|
|      |
|      +---1--- IPsecTunnelAction // set up SA from host to
|      |                                     // gateway1
|      |
|      +---2--- IPsecTunnelAction // or set up SA to gateway2
|
+---2--- IPsecTransportAction      // then set up SA from host
```

```
// through tunnel to remote
// host
```

In the case of "Do All", a couple of actions can be executed successfully before a subsequent action fails. In this case, some IKE or IPsec actions may have resulted in SAs creation. Even if the net effect of the aggregated actions is failure, those created SAs MAY be kept or MAY be deleted.

In the case of "Do All", the IPsec selectors to be used during IPsec SA negotiation are:

- for the last IPsecAction of the aggregation (i.e. usually the innermost IPsec SA): this is the combination of the IPHeadersFilter class and of the Granularity property of the IPsecAction;
- for all other IPsecActions of the aggregation: the selector is the source IP address being the local IP address and the destination IP address being the PeerGateway IP address of the following IPsecAction of the "Do All" aggregation. NB: the granularity is IP address to IP address.

If the above behavior is not desirable, the alternative is to define several SARules one for each IPsec SA to be built. This will allow the definition of specific IPsec selectors for all IPsecActions.

4.2.3 The Property LimitNegotiation

The property LimitNegotiation is used as part of processing either an IKE or an IPsec rule.

Before proceeding with a phase 1 negotiation, this property is checked to determine if the negotiation role of the rule matches that defined for the negotiation being undertaken (e.g., Initiator, Responder, or Both). If this check fails (e.g. the current role is IKE responder while the rule specifies IKE initiator), then the IKE negotiation is stopped. Note that this only applies to new IKE phase 1 negotiations and has no effect on either renegotiation or refresh operations with peers for which an established SA already exists.

Before proceeding with a phase 2 negotiation, the LimitNegotiation property of the IPsecRule is first checked to determine if the negotiation role indicated for the rule matches that of the current negotiation (Initiator, Responder, or Either). Note that this limit applies only to new phase 2 negotiations. It is ignored when an attempt is made to refresh an expiring SA (either side can initiate a refresh operation). The IKE system can determine that the negotiation is a refresh operation by checking to see if the selector information matches that of an existing SA. If LimitNegotiation does not match and the selector corresponds to a new SA, the negotiation

is stopped.

The property is defined as follows:

NAME	LimitNegotiation
DESCRIPTION	Limits the role to be undertaken during negotiation.
SYNTAX	unsigned 16-bit integer
VALUE	1 - initiator-only
	2 - responder-only
	3 - both

[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, plus
               IdentityContexts
```

4.3.1. The Property IdentityContexts

The IKE service of a security endpoint may have multiple identities for use in different situations. The combination of the interface (represented by the `IPProtocolEndpoint` or by a collection of `IPProtocolEndpoints`), the identity type (as specified in the `IKEAction`) and the `IdentityContexts` specifies a unique identity.

The `IdentityContexts` property specifies the context to select the relevant IKE identity to be used during the further `IKEAction`. A context may be a VPN name or other identifier for selecting the appropriate identity for use on the protected `IPProtocolEndpoint` (or collection of `IPProtocolEndpoints`).

`IdentityContexts` is an array of strings. The multiple values in the array are logically OR'ed together in evaluating the `IdentityContexts`. Each value in the array may be the composition of multiple context names. So, a single value may be a single context name (e.g., "CompanyXVPN") or it may be combination of contexts. When an array value is a composition, the individual values are logically AND'ed together for evaluation purposes and the syntax is:

```
<ContextName>[&&<ContextName>]*
```

where the individual context names appear in alphabetical order (according to the collating sequence for UCS-2). So, for example, the values "CompanyXVPN", "CompanyYVPN&&TopSecret", "CompanyZVPN&&Confidential" means that, for the appropriate `IPProtocolEndpoint` and `IdentityType`, the contexts are matched if the identity specifies "CompanyXVPN" or "CompanyYVPN&&TopSecret" or "CompanyZVPN&&Confidential".

The property is defined as follows:

```
NAME          IdentityContexts
DESCRIPTION    Specifies the context in which to select the IKE
               identity.
SYNTAX         string array
```

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          IPsecRule
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 Association Class IPsecPolicyForEndpoint

The class IPsecPolicyForEndpoint associates an IPsecPolicyGroup with a specific network interface. If an IPProtocolEndpoint of a system does not have an IPsecPolicyForEndpoint-associated IPsecPolicyGroup, then the IPsecPolicyForSystem associated IPsecPolicyGroup is used for that endpoint. The class definition for IPsecPolicyForEndpoint is as follows:

NAME IPsecPolicyForEndpoint
DESCRIPTION Associates a policy group to a network interface.
DERIVED FROM Dependency (see [[CIMCORE](#)])
ABSTRACT FALSE
PROPERTIES Antecedent[ref IPProtocolEndpoint[0..n]]
 Dependent[ref IPsecPolicyGroup[0..1]]

4.5.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to an IPProtocolEndpoint instance. The [0..n] cardinality indicates that an IPsecPolicyGroup instance may be associated with zero or more IPProtocolEndpoint instances.

4.5.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to an IPsecPolicyGroup instance. The [0..1] cardinality indicates that an IPProtocolEndpoint instance may have an association to at most one IPsecPolicyGroup instance.

4.6. The Association Class IPsecPolicyForSystem

The class IPsecPolicyForSystem associates an IPsecPolicyGroup with a specific system. If an IPProtocolEndpoint of a system does not have an IPsecPolicyForEndpoint-associated IPsecPolicyGroup, then the IPsecPolicyForSystem associated IPsecPolicyGroup is used for that endpoint. The class definition for IPsecPolicyForSystem is as follows:

NAME IPsecPolicyForSystem
DESCRIPTION Default policy group for a system.
DERIVED FROM Dependency (see [[CIMCORE](#)])
ABSTRACT FALSE
PROPERTIES Antecedent[ref System[0..n]]
 Dependent[ref IPsecPolicyGroup[0..1]]

4.6.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to a System instance. The [0..n] cardinality indicates that an IPsecPolicyGroup instance may have an association to zero or more System instances.

4.6.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to an IPsecPolicyGroup instance. The [0..1] cardinality indicates that a System instance may have an association to at most one IPsecPolicyGroup instance.

4.7. The Aggregation Class SARuleInPolicyGroup

The class SARuleInPolicyGroup associates a SARule with the IPsecPolicyGroup that contains it. The class definition for SARuleInPolicyGroup is as follows:

```
NAME          SARuleInPolicyGroup
DESCRIPTION    Associates a SARule with the IPsecPolicyGroup that
               contains it.
DERIVED FROM  PolicySetComponent (see [PCIME])
ABSTRACT      FALSE
PROPERTIES    Priority (from PolicySetComponent)
               GroupComponent [ref IPsecPolicyGroup [0..n]]
               PartComponent [ref SARule [0..n]]
```

Note: an implementation can easily partition the set of SARules aggregated by a SARuleInPolicyGroup instance into one IKERule instances subset and into one IPsecRule instances subset based on the class type of the component instances (being either IKERule or IPsecRule instances).

4.7.1. The Property Priority

For a description of this property, see [[PCIME](#)].

4.7.2. The Reference GroupComponent

The property GroupComponent is inherited from PolicyRuleInPolicyGroup and is overridden to refer to an IPsecPolicyGroup instance. The [0..n] cardinality indicates that a SARule instance may be shared across multiple IPsecPolicyGroups).

4.7.3. The Reference PartComponent

The property PartComponent is inherited from PolicyRuleInPolicyGroup and is overridden to refer to a SARule instance. The [0..n] cardinality indicates that an IPsecPolicyGroup instance may contain zero or more SARule instances.

4.8. The Aggregation Class SAConditionInRule

The class SAConditionInRule associates an SARule with the SACondition instance(s) that trigger(s) it. The class definition for SAConditionInRule is as follows:

```
NAME          SAConditionInRule
DESCRIPTION    Associates an SARule with the SACondition instance(s)
               that trigger(s) it.
DERIVED FROM  PolicyConditionInPolicyRule (see [PCIM] & [PCIME])
ABSTRACT      FALSE
```

PROPERTIES GroupNumber (from PolicyConditionInPolicyRule)
 ConditionNegated (from PolicyConditionInPolicyRule)
 GroupComponent [ref SARule [0..n]]
 PartComponent [ref SACondition [1..n]]

4.8.1. The Properties GroupNumber and ConditionNegated

For a description of these properties, see [[PCIM](#)].

4.8.2. The Reference GroupComponent

The property `GroupComponent` is inherited from `PolicyConditionInPolicyRule` and is overridden to refer to an `SARule` instance. The `[0..n]` cardinality indicates that an `SACondition` instance may be contained in zero or more `SARule` instances.

Note: the 0 cardinality allows `SACondition` instances to exist without being contained in a `SARule`.

4.8.3. The Reference PartComponent

The property `PartComponent` is inherited from `PolicyConditionInPolicyRule` and is overridden to refer to an `SACondition` instance. The `[1..n]` cardinality indicates that an `SARule` instance MUST contain at least one `SACondition` instance.

4.9. The Aggregation Class PolicyActionInSARule

The `PolicyActionInSARule` class associates an `SARule` with one or more `PolicyAction` instances. In all cases where an `SARule` is being used, the contained actions MUST be either subclasses of `SAAction` or instances of `CompoundPolicyAction`. For an `IKERule`, the contained actions MUST be related to phase 1 processing, i.e., `IKEAction` or `IKERejectAction`. Similarly, for an `IPsecRule`, contained actions MUST be related to phase 2 or preconfigured SA processing, e.g., `IPsecTransportAction`, `IPsecBypassAction`, etc. The class definition for `PolicyActionInSARule` is as follows:

NAME	<code>PolicyActionInSARule</code>
DESCRIPTION	Associates an <code>SARule</code> with its <code>PolicyAction(s)</code> .
DERIVED FROM	<code>PolicyActionInPolicyRule</code> (see [PCIM] & [PCIMe])
ABSTRACT	FALSE
PROPERTIES	<code>GroupComponent</code> [ref <code>SARule</code> <code>[0..n]</code>] <code>PartComponent</code> [ref <code>PolicyAction</code> <code>[1..n]</code>] <code>ActionOrder</code> (from <code>PolicyActionInPolicyRule</code>)

4.9.1. The Reference GroupComponent

The property `GroupComponent` is inherited from `PolicyActionInPolicyRule` and is overridden to refer to an `SARule` instance. The `[0..n]` cardinality indicates that an `SAAction` instance may be contained in zero or more `SARule` instances.

4.9.2. The Reference PartComponent

The property `PartComponent` is inherited from `PolicyActionInPolicyRule` and is overridden to refer to an `SAAction` or `CompoundPolicyAction` instance. The `[1..n]` cardinality indicates that an `SARule` instance MUST contain at least one `SAAction` or `CompoundPolicyAction` instance.

4.9.3. The Property ActionOrder

The property ActionOrder is inherited from the superclass PolicyActionInPolicyRule. It specifies the relative position of this PolicyAction in the sequence of actions associated with a PolicyRule. The ActionOrder MUST be unique so as to provide a deterministic order. In addition, the actions in an SARule are executed as follows. See [section 4.2.2](#) ExecutionStrategy for a discussion on the use of the ActionOrder property.

The property is defined as follows:

NAME	ActionOrder
------	-------------

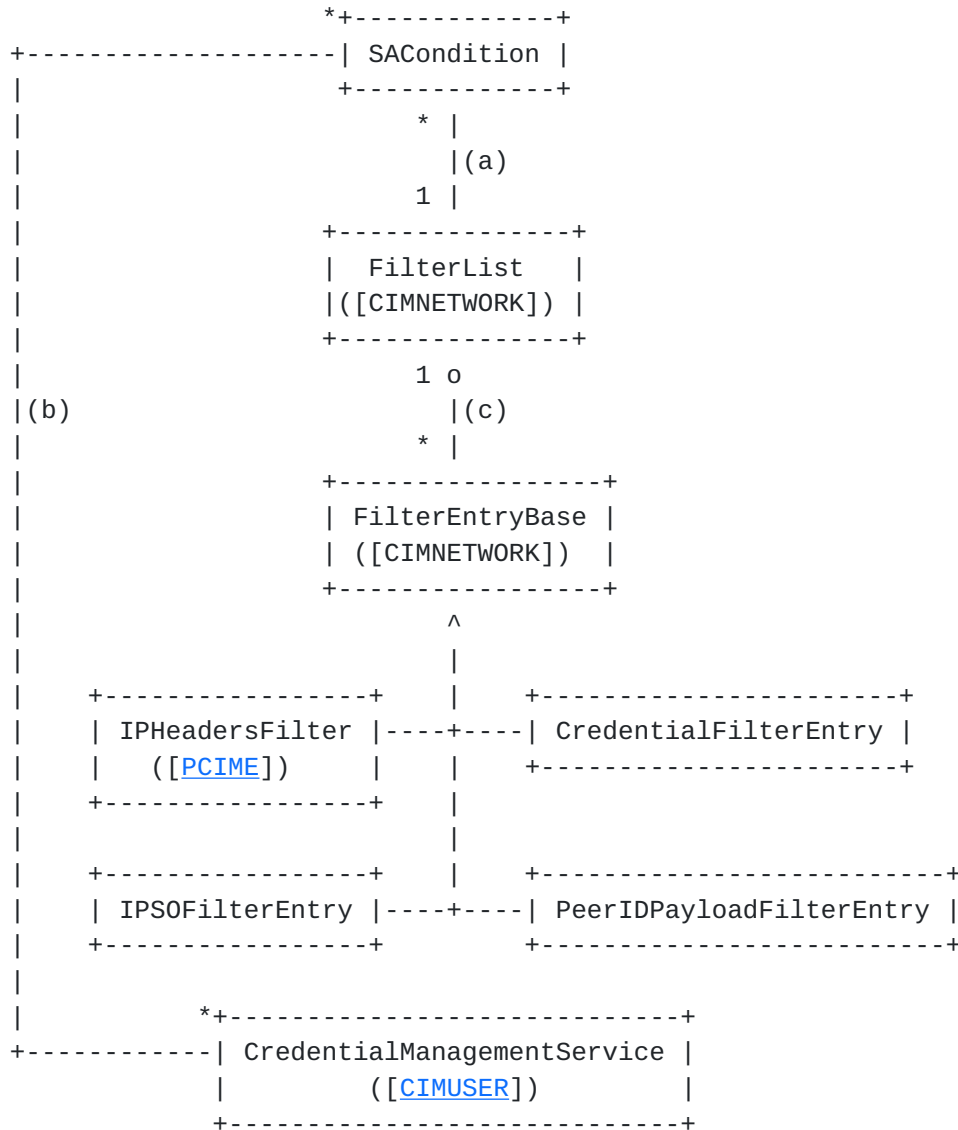
DESCRIPTION Specifies the order of actions.

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 SAActions with the same precedence 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.



(a) FilterOfSACondition

(b) AcceptCredentialsFrom

(c) EntriesInFilterList (see [CIMNETWORK])

5.1. The Class SACondition

The class SACondition defines the conditions of rules for IKE and IPsec negotiations. Conditions are associated with policy rules via the SAConditionInRule aggregation. It is used as an anchor point to associate various types of filters with policy rules via the

FilterOfSACondition association. It also defines whether Credentials can be accepted for a particular policy rule via the AcceptCredentialsFrom association.

Associated objects represent components of the condition that may or may not apply at a given rule evaluation. For example, an AcceptCredentialsFrom evaluation is only performed when a credential is available to be evaluated against the list of trusted credential management services. Similarly, a PeerIDPayloadFilterEntry may only be evaluated when an IDPayload value is available to compare with the filter. Condition components that do not have corresponding values with which to evaluate are evaluated as TRUE unless the protocol has completed without providing the required information.

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)
```

5.2. The Class IPHeadersFilter

The class IPHeadersFilter is defined in [[PCIMe](#)] with the following note:

- 1) to specify 5-tuple filters that are to apply symmetrically (i.e., matches traffic in both directions of the same flows which is quite typical for SPD entries for ingress and egress traffic), the Direction property of the FilterList SHOULD be set to "Mirrored".

5.3. The Class CredentialFilterEntry

The class CredentialFilterEntry defines an equivalence class that match credentials of IKE peers. Each CredentialFilterEntry includes a MatchFieldName that is interpreted according to the CredentialManagementService(s) associated with the SACondition (AcceptCredentialsFrom).

These credentials can be X.509 certificates, Kerberos tickets, or other types of credentials obtained during the Phase 1 exchange.

Note: this filter entry will probably be checked while the IKE negotiation takes place. If the check is a failure, then the IKE negotiation MUST be stopped, and the result of the IKEAction which triggered this negotiation is a failure.

The class definition for CredentialFilterEntry is as follows:

```
NAME          CredentialFilterEntry
DESCRIPTION   Specifies a match filter based on the IKE credentials.
DERIVED FROM  FilterEntryBase (see [CIMNETWORK])
ABSTRACT      FALSE
PROPERTIES   Name (from FilterEntryBase)
              IsNegated (from FilterEntryBase)
              MatchFieldName
              MatchFieldValue
              CredentialType
```

5.3.1. The Property MatchFieldName

The property MatchFieldName specifies the sub-part of the credential to match against MatchFieldValue. The property is defined as follows:

NAME	MatchFieldName
DESCRIPTION	Specifies which sub-part of the credential to match.
SYNTAX	string
VALUE	This is the string representation of a X.509 certificate attribute, e.g.: <ul style="list-style-type: none">- "serialNumber"- "signatureAlgorithm"

- "issuerName"
- "subjectName"
- "subjectAltName"
- ...

5.3.2. The Property MatchFieldValue

The property MatchFieldValue specifies the value to compare with the MatchFieldName in a credential to determine if the credential matches this filter entry. The property is defined as follows:

NAME	MatchFieldValue
DESCRIPTION	Specifies the value to be matched by the MatchFieldName.
SYNTAX	string
VALUE	NB: If the CredentialFilterEntry corresponds to a DistinguishedName, this value in the CIM class is represented by an ordinary string value. However, an implementation must convert this string to a DER-encoded string before matching against the values extracted from credentials at runtime.

A wildcard mechanism can be used in the MatchFieldValue string. E.g., if the MatchFieldName is "subjectName" then a MatchFieldValue of "cn=*,ou=engineering,o=foo,c=be" will match successfully a certificate whose subject attribute is "cn=Jane Doe,ou=engineering,o=foo,c=be". The wildcard character '*' can be used to represent 0 or several characters.

5.3.3. The Property CredentialType

The property CredentialType specifies the particular type of credential that is being matched. The property is defined as follows:

NAME	CredentialType
DESCRIPTION	Defines the type of IKE credentials.
SYNTAX	unsigned 16-bit integer
VALUE	1 - X.509 Certificate 2 - Kerberos Ticket

5.4. The Class IPSOFilterEntry

The class IPSOFilterEntry is used to match traffic based on the IP Security Options header values (ClassificationLevel and ProtectionAuthority) as defined in [RFC1108](#). This type of filter entry is used to adjust the IPsec encryption level according to the IPSO classification of the traffic (e.g., secret, confidential, restricted, etc. The class definition for IPSOFilterEntry is as follows:

NAME IPSOFilterEntry
DESCRIPTION Specifies the a match filter based on IP Security
Options.
DERIVED FROM FilterEntryBase (see [CIMNETWORK])
ABSTRACT FALSE
PROPERTIES Name (from FilterEntryBase)
IsNegated (from FilterEntryBase)
MatchConditionType
MatchConditionValue

[5.4.1.](#) The Property MatchConditionType

The property `MatchConditionType` specifies the IPSO header field that will be matched (e.g., traffic classification level or protection authority). The property is defined as follows:

NAME	<code>MatchConditionType</code>
DESCRIPTION	Specifies the IPSO header field to be matched.
SYNTAX	unsigned 16-bit integer
VALUE	1 - <code>ClassificationLevel</code> 2 - <code>ProtectionAuthority</code>

5.4.2. The Property `MatchConditionValue`

The property `MatchConditionValue` specifies the value of the IPSO header field to be matched against. The property is defined as follows:

NAME	<code>MatchConditionValue</code>
DESCRIPTION	Specifies the value of the IPSO header field to be matched against.
SYNTAX	unsigned 16-bit integer
VALUE	The values MUST be one of values listed in RFC 1108 (or any further IANA Assigned Numbers document). Some examples for <code>ClassificationLevel</code> are: 61 - <code>TopSecret</code> 90 - <code>Secret</code> 150 - <code>Confidential</code> 171 - <code>Unclassified</code> For <code>ProtectionAuthority</code> , some examples are: 0 - <code>GENSER</code> 1 - <code>SIOP-ESI</code> 2 - <code>SCI</code> 3 - <code>NSA</code> 4 - <code>DOE</code>

5.5. The Class `PeerIDPayloadFilterEntry`

The class `PeerIDPayloadFilterEntry` defines filters used to match ID payload values from the IKE protocol exchange. `PeerIDPayloadFilterEntry` permits the specification of certain ID payload values such as `"*@company.com"` or `"193.190.125.0/24"`.

Obviously this filter applies only to `IKERules` when acting as a responder. Moreover, this filter can be applied immediately in the case of aggressive mode but its application is to be delayed in the case of main mode. The class definition for `PeerIDPayloadFilterEntry` is as follows:

NAME	<code>PeerIDPayloadFilterEntry</code>
DESCRIPTION	Specifies a match filter based on IKE identity.

DERIVED FROM FilterEntryBase (see [CIMNETWORK])
ABSTRACT FALSE
PROPERTIES Name (from FilterEntryBase)
IsNegated (from FilterEntryBase)
MatchIdentityType
MatchIdentityValue

5.5.1. The Property MatchIdentityType

The property MatchIdentityType specifies the type of identity provided by the peer in the ID payload. The property is defined as follows:

NAME	MatchIdentityType
DESCRIPTION	Specifies the ID payload type.
SYNTAX	unsigned 16-bit integer
VALUE	Consult [DOI] for valid values.

5.5.2. The Property MatchIdentityValue

The property MatchIdentityValue specifies the filter value for comparison with the ID payload, e.g., "*@company.com". The property is defined as follows:

NAME	MatchIdentityValue
DESCRIPTION	Specifies the ID payload value.
SYNTAX	string
VALUE	NB: The syntax may need to be converted for comparison. If the PeerIDPayloadFilterEntry type is a DistinguishedName, the name in the MatchIdentityValue property is represented by an ordinary string value, but this value must be converted into a DER-encoded string before matching against the values extracted from IKE ID payloads at runtime. The same applies to IPv4 & IPv6 addresses.

Different wildcard mechanisms can be used depending on the ID payload:

- a MatchIdentityValue of "*@company.com" will match a user FQDN ID payload of "JDOE@COMPANY.COM"
- a MatchIdentityValue of "*.company.com" will match a FQDN ID payload of "WWW.COMPANY.COM"
- a MatchIdentityValue of "cn=*,ou=engineering,o=company,c=us" will match a DER DN ID payload of "cn=John Doe,ou=engineering,o=company,c=us"
- a MatchIdentityValue of "193.190.125.0/24" will match an IPv4 address ID payload of 193.190.125.10
- a MatchIdentityValue of "193.190.125.*" will also match an IPv4 address ID payload of 193.190.125.10.

The above wildcard mechanisms MUST be supported for all ID payloads supported by the local IKE entity. The character '*' replaces 0 or multiple instances of any character.

5.6. The Association 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 makes
              up the individual condition elements.
DERIVED FROM  Dependency (see [CIMCORE])
ABSTRACT      FALSE
PROPERTIES    Antecedent [ref FilterList[1..1]]
              Dependent  [ref SACondition[0..n]]
```

[5.6.1](#). The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to a FilterList instance. The [1..1] cardinality indicates that an SACondition instance MUST be associated with one and only one FilterList instance.

5.6.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to an SACondition instance. The [0..n] cardinality indicates that a FilterList instance may be associated with zero or more SACondition instances.

5.7. The Association Class AcceptCredentialFrom

The class AcceptCredentialFrom specifies which credential management services (e.g., a CertificateAuthority or a Kerberos service) are to be trusted to certify peer credentials. This is used to assure that the credential being matched in the CredentialFilterEntry is a valid credential that has been supplied by an approved CredentialManagementService. If a CredentialManagementService is specified and a corresponding CredentialFilterEntry is used, but the credential supplied by the peer is not certified by that CredentialManagementService (or one of the CredentialManagementServices in its trust hierarchy), the CredentialFilterEntry is deemed not to match. If a credential is certified by a CredentialManagementService in the AcceptCredentialsFrom list of services but there is no CredentialFilterEntry, this is considered equivalent to a CredentialFilterEntry that matches all credentials from those services.

The class definition for AcceptCredentialFrom is as follows:

NAME	AcceptCredentialFrom
DESCRIPTION	Associates a condition with the credential management services to be trusted.
DERIVED FROM	Dependency (see [CIMCORE])
ABSTRACT	FALSE
PROPERTIES	Antecedent [ref CredentialManagementService[0..n]] Dependent [ref SACondition[0..n]]

5.7.1. The Reference Antecedent

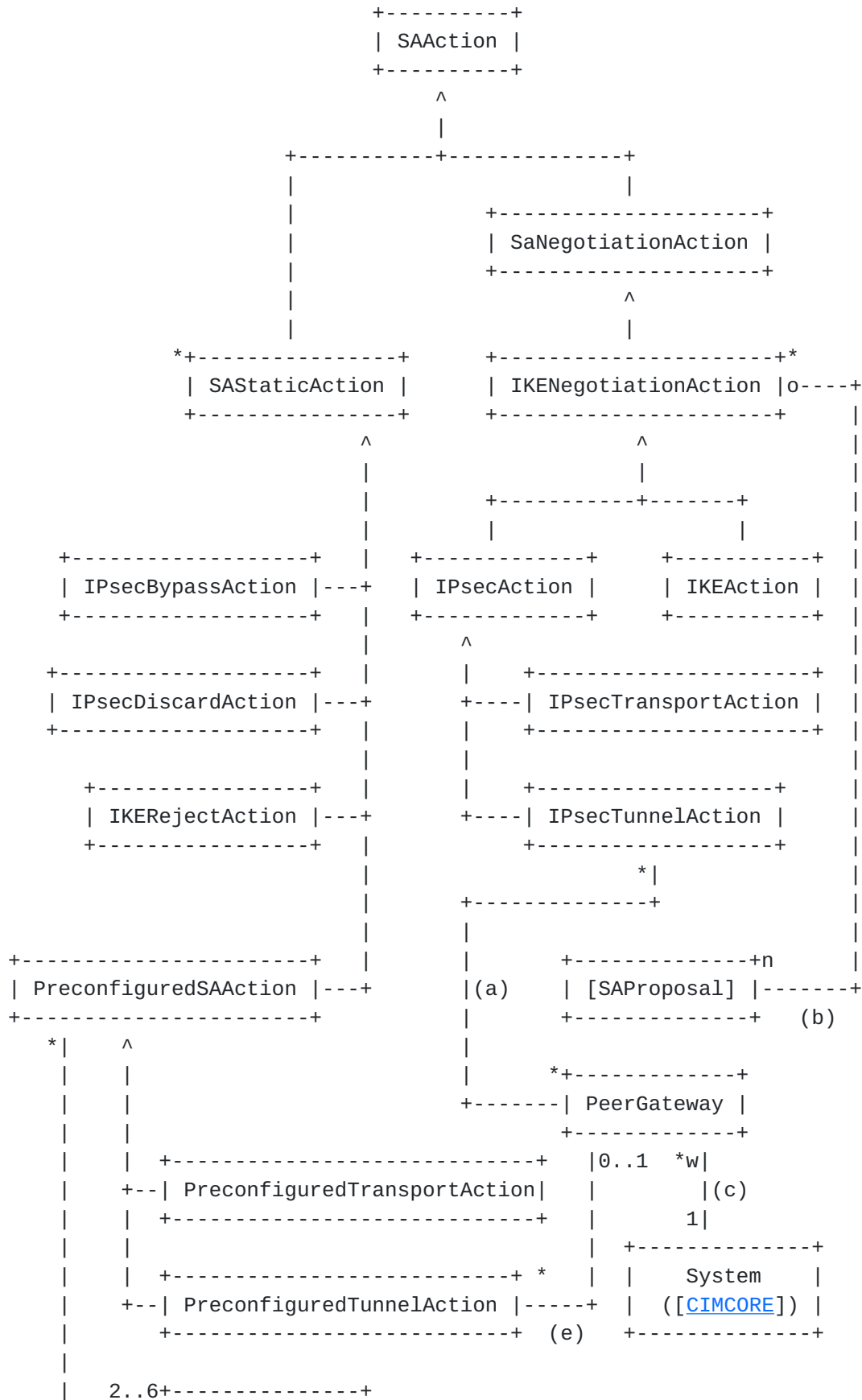
The property Antecedent is inherited from Dependency and is overridden to refer to a CredentialManagementService instance. The [0..n] cardinality indicates that an SACondition instance may be associated with zero or more CredentialManagementService instances.

5.7.2. The Reference Dependent

The property `Dependent` is inherited from `Dependency` and is overridden to refer to a `SACondition` instance. The `[0..n]` cardinality indicates that a `CredentialManagementService` instance may be associated with zero or more `SACondition` instances.

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.



```
+-----| [SATransform] |  
(d)  +-----+
```

- (a) PeerGatewayForTunnel
- (b) ContainedProposal
- (c) HostedPeerGatewayInformation
- (d) TransformOfPreconfiguredAction
- (e) PeerGatewayForPreconfiguredTunnel

6.1. The Class SAAction

The class SAAction is abstract and serves as the base class for IKE and IPsec actions. It is used for aggregating different types of

actions to IKE and IPsec rules. 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 TRUE
PROPERTIES PolicyActionName (from PolicyAction)
 DoActionLogging
 DoPacketLogging

6.1.1. The Property DoActionLogging

The property DoActionLogging specifies whether a log message is to be generated when the action is performed. This applies for SANegotiationActions with the meaning of logging a message when the negotiation is attempted (with the success or failure result). This also applies for SASStaticAction only for PreconfiguredSAAction with the meaning of logging a message when the preconfigured SA is actually installed in the SADB. The property is defined as follows:

NAME DoActionLogging
DESCRIPTION Specifies the whether to log when the action is performed.
SYNTAX boolean
VALUE true - a log message is to be generated when action is performed.
 false - no log message is to be generated when action is performed.

6.1.2. The Property DoPacketLogging

The property DoPacketLogging specifies whether a log message is to be generated when the resulting security association is used to process the packet. If the SANegotiationAction successfully executes and results in the creation of one or several security associations or if the PreconfiguredSAAction executes, the value of DoPacketLogging SHOULD be propagated to an optional field of SADB. This optional field should be used to decide whether a log message is to be generated when the SA is used to process a packet. For SASStaticActions, a log message is to be generated when the IPsecBypassAction, IPsecDiscardAction, IKERjectAction are executed. The property is defined as follows:

NAME DoPacketLogging
DESCRIPTION Specifies the whether to log when the resulting security association is used to process the packet.
SYNTAX boolean

VALUE true - a log message is to be generated when the
 resulting security association is used to process the
 packet.
 false - no log message is to be generated.

6.2. The Class SStaticAction

The class SStaticAction is abstract and serves as the base class for IKE and IPsec actions that do not require any negotiation. The class definition for SStaticAction is as follows:

NAME SStaticAction

DESCRIPTION The base class for IKE and IPsec actions that do not require any negotiation.

DERIVED FROM SAAction

ABSTRACT TRUE

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 non-zero value is typically used in conjunction with alternate SAActions performed when there is a negotiation failure of some sort.

Note: if the referenced SASStaticAction object is a PreconfiguredSAAction associated to several SATransforms, then the actual lifetime of the preconfigured SA will be the lesser of the value of this LifetimeSeconds property and of the value of the MaxLifetimeSeconds property of the associated SATransform. If the value of this LifetimeSeconds property is zero, then there will be no lifetime associated to this SA.

It is expected that most SASStaticAction instances will have their LifetimeSeconds properties set to zero (meaning no expiration of the resulting SA).

6.3. The Class IPsecBypassAction

The class IPsecBypassAction is used when packets are allowed to be processed without applying IPsec encapsulation 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 SASStaticAction

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 SStaticAction
ABSTRACT      FALSE
```

[6.5.](#) The Class IKERejectAction

The class `IKERjectAction` is used to prevent attempting an IKE negotiation with the peer(s). The main use of this class is to prevent some denial of service attacks when acting as IKE responder. It goes beyond a plain discard of UDP/500 IKE packets because the `SACondition` can be based on specific `PeerIDPayloadFilterEntry` (when aggressive mode is used). The class definition for `IKERjectAction` is as follows:

```
NAME           IKERjectAction
DESCRIPTION    Specifies that an IKE negotiation should not even be
               attempted or continued.
DERIVED FROM  SStaticAction
ABSTRACT      FALSE
```

6.6. The Class PreconfiguredSAAction

The class `PreconfiguredSAAction` is used to create a security association using preconfigured, hard-wired algorithms and keys.

Notes:

- the SPI for a `PreconfiguredSAAction` is contained in the association, `TransformOfPreconfiguredAction`;
- the session key (if applicable) is contained in an instance of the class `SharedSecret` (see [[CIMUSER](#)]). The session key is stored in the property `Secret`, the property `protocol` contains either "ESP-encrypt", "ESP-auth" or "AH", the property `algorithm` contains the algorithm used to protect the secret (can be "PLAINTEXT" if the IPsec entity has no secret storage), the value of property `RemoteID` is the concatenation of the remote IPsec peer IP address in dotted decimal, of the character "/", of "IN" (respectively "OUT") for inbound SA (respectively outbound SA), of the character "/" and of the hexadecimal representation of the SPI.

Although the class is concrete, it MUST not be instantiated. The class definition for `PreconfiguredSAAction` is as follows:

```
NAME           PreconfiguredSAAction
DESCRIPTION    Specifies preconfigured algorithm and keying information
               for creation of a security association.
DERIVED FROM  SStaticAction
ABSTRACT      TRUE
PROPERTIES    LifetimeKilobytes
```

6.6.1. The Property LifetimeKilobytes

The property `LifetimeKilobytes` specifies a traffic limit in kilobytes that can be consumed before the SA is deleted.. The property is

defined as follows:

NAME	LifetimeKilobytes
DESCRIPTION	Specifies the SA lifetime in kilobytes.
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 non-zero value is used to indicate that after this number of kilobytes has been consumed the SA must be deleted from the SADB.

Note: the actual lifetime of the preconfigured SA will be the lesser of the value of this LifetimeKilobytes property and of the value of the MaxLifetimeSeconds property of the associated SATransform. If the value of this LifetimeKilobytes property is zero, then there will be no lifetime associated with this action.

It is expected that most PreconfiguredSAAction instances will have their LifetimeKilobyte properties set to zero (meaning no expiration of the resulting SA).

6.7. The Class PreconfiguredTransportAction

The class PreconfiguredTransportAction is used to create an IPsec transport-mode security association using preconfigured, hard-wired algorithms and keys. The class definition for PreconfiguredTransportAction is as follows:

NAME PreconfiguredTransportAction
DESCRIPTION Specifies preconfigured algorithm and keying information
 for creation of an IPsec transport security association.
DERIVED FROM PreconfiguredSAAction
ABSTRACT FALSE

6.8. The Class PreconfiguredTunnelAction

The class PreconfiguredTunnelAction is used to create an IPsec tunnel-mode security association using preconfigured, hard-wired algorithms and keys. The class definition for PreconfiguredSAAction is as follows:

NAME PreconfiguredTunnelAction
DESCRIPTION Specifies preconfigured algorithm and keying information
 for creation of an IPsec tunnel-mode security
 association.
DERIVED FROM PreconfiguredSAAction
ABSTRACT FALSE
PROPERTIES DFHandling

6.8.1. The Property DFHandling

The property DFHandling specifies how the Don't Fragment bit of the internal IP header is to be handled during IPsec processing. The property is defined as follows:

NAME DFHandling
DESCRIPTION Specifies the processing of the DF bit.
SYNTAX unsigned 16-bit integer
VALUE 1 - Copy the DF bit from the internal IP header to the
 external IP header.
 2 - Set the DF bit of the external IP header to 1.

3 - Clear the DF bit of the external IP header to 0.

6.9. The Class SANegotiationAction

The class SANegotiationAction specifies an action requesting security policy negotiation.

This is an abstract class. Currently, only one security policy negotiation protocol action is subclassed from SANegotiationAction: the IKENegotiationAction class. It is nevertheless expected that other security policy negotiation protocols will exist and the

negotiation actions of those new protocols would be modeled as a subclass of SANegotiationAction.

NAME SANegotiationAction
 DESCRIPTION Specifies a negotiation action .
 DERIVED FROM SAAction
 ABSTRACT TRUE

6.10. The Class IKENegotiationAction

The class IKENegotiationAction is abstract and serves as the base class for IKE and IPsec actions that result in a IKE negotiation. The class definition for IKENegotiationAction is as follows:

NAME IKENegotiationAction
 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 SANegotiationAction
 ABSTRACT TRUE
 PROPERTIES MinLifetimeSeconds
 MinLifetimeKilobytes
 IdleDurationSeconds

6.10.1. 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 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.

6.10.2. The Property MinLifetimeKilobytes

The property MinLifetimeKilobytes specifies the minimum kilobytes 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. Note that there has been considerable debate regarding the usefulness of applying kilobyte lifetimes to IKE phase 1 security associations, so

it is likely that this property will only apply to the sub-class IPsecAction. The property is defined as follows:

NAME	MinLifetimeKilobytes
DESCRIPTION	Specifies the minimum acceptable kilobytes 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 kilobytes lifetime.

6.10.3. 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	<code>IdleDurationSeconds</code>
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 (only the seconds and kilobyte lifetimes will be used). Any non-zero value indicates the number of seconds the security association may remain unused.

6.11. 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. The class definition for `IPsecAction` is as follows:

NAME	<code>IPsecAction</code>
DESCRIPTION	A base class for IPsec transport and tunnel actions that specifies the parameters for IKE phase 2 IPsec DOI negotiations.
DERIVED FROM	<code>IKENegotiationAction</code>
ABSTRACT	TRUE
PROPERTIES	<code>UsePFS</code> <code>UseIKEGroup</code> <code>GroupId</code> <code>Granularity</code> <code>VendorID</code>

6.11.1. 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	<code>UsePFS</code>
DESCRIPTION	Specifies the whether or not to use PFS when refreshing keys.
SYNTAX	boolean
VALUE	A value of true indicates that PFS should be used. A value of false indicates that PFS should not be used.

6.11.2. The Property `UseIKEGroup`

The property `UseIKEGroup` specifies whether or not phase 2 should use

the same key exchange group 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 key exchange group to use for phase 2.

6.11.3. The Property GroupId

The property GroupId specifies the key exchange 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. If the GroupID number is from the vendor-specific range (32768-65535), the property VendorID qualifies the group number. The property is defined as follows:

NAME	GroupId
DESCRIPTION	Specifies the key exchange group to use for phase 2 when the property UsePFS is true and the property UseIKEGroup is false.
SYNTAX	unsigned 16-bit integer
VALUE	Consult [IKE] for valid values.

6.11.4. The Property Granularity

The property Granularity specifies how the selector for the security association should be derived from the traffic that triggered the negotiation. The property is defined as follows:

NAME	Granularity
DESCRIPTION	Specifies the how the proposed selector for the security association will be created.
SYNTAX	unsigned 16-bit integer
VALUE	1 - subnet: the source and destination subnet masks of the filter entry are used. 2 - address: only the source and destination IP addresses of the triggering packet are used. 3 - protocol: the source and destination IP addresses and the IP protocol of the triggering packet are used. 4 - port: the source and destination IP addresses and the IP protocol and the source and destination layer 4 ports of the triggering packet are used.

6.11.5. The Property VendorID

The property VendorID is used together with the property GroupID (when it is in the vendor-specific range) to identify the key exchange group. VendorID is ignored unless UsePFS is true and UseIKEGroup is false and GroupID is in the vendor-specific range (32768-65535). The property is defined as follows:

NAME	VendorID
DESCRIPTION	Specifies the IKE Vendor ID.
SYNTAX	string

6.12. 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
```

[6.13](#). 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    DFHandling
```

6.13.1. The Property DFHandling

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

```
NAME          DFHandling
DESCRIPTION    Specifies how to process the DF bit.
SYNTAX        unsigned 16-bit integer
VALUE         1 - Copy the DF bit from the internal IP header to the
               external IP header.
               2 - Set the DF bit of the external IP header to 1.
               3 - Clear the DF bit of the external IP header to 0.
```

6.14. 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  IKENegotiationAction
ABSTRACT      FALSE
PROPERTIES    ExchangeMode
               UseIKEIdentityType
               VendorID
               AggressiveModeGroupId
```

6.14.1. The Property ExchangeMode

The property `ExchangeMode` specifies which IKE mode should be used for IKE phase 1 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
```

6.14.2. 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 with the IKE identities available on the system and the IdentityContexts of the matching IKERule. The property is defined as follows:

NAME	UseIKEIdentityType
DESCRIPTION	Specifies the IKE identity to use during negotiation.

SYNTAX unsigned 16-bit integer
VALUE Consult [[DOI](#)] for valid values.

6.14.3. The Property VendorID

The property VendorID specifies the value to be used in the Vendor ID payload. The property is defined as follows:

NAME VendorID
DESCRIPTION Vendor ID Payload.
SYNTAX string
VALUE A value of NULL means that Vendor ID payload will be
 neither generated nor accepted. A non-NULL value means
 that a Vendor ID payload will be generated (when acting
 as an initiator) or is expected (when acting as a
 responder).

6.14.4. The Property AggressiveModeGroupId

The property AggressiveModeGroupId specifies which group ID is to be used in the first packets of the phase 1 negotiation. This property is ignored unless the property ExchangeMode is set to 4 (aggressive mode). If the AggressiveModeGroupId number is from the vendor-specific range (32768-65535), the property VendorID qualifies the group number. The property is defined as follows:

NAME AggressiveModeGroupId
DESCRIPTION Specifies the group ID to be used for aggressive mode.
SYNTAX unsigned 16-bit integer

6.15. The Class PeerGateway

The class PeerGateway specifies the security gateway with which the IKE services negotiates. The class definition for PeerGateway is as follows:

NAME PeerGateway
DESCRIPTION Specifies the security gateway with which to negotiate.
DERIVED FROM LogicalElement (see [[CIMCORE](#)])
ABSTRACT FALSE
PROPERTIES Name
 PeerIdentityType
 PeerIdentity

Note: the class PeerIdentityEntry contains more information about the peer (namely its IP address).

6.15.1. The Property Name

The property Name specifies a user-friendly name for this security

gateway. The property is defined as follows:

NAME	Name
DESCRIPTION	Specifies a user-friendly name for this security gateway.
SYNTAX	string

6.15.2. The Property PeerIdentityType

The property PeerIdentityType specifies the IKE identity type of the security gateway. The property is defined as follows:

NAME	PeerIdentityType
DESCRIPTION	Specifies the IKE identity type of the security gateway.
SYNTAX	unsigned 16-bit integer
VALUE	Consult [DOI] for valid values.

6.15.3. The Property PeerIdentity

The property PeerIdentity specifies the IKE identity value of the security gateway. A conversion may be needed between the PeerIdentity string representation and the real value used in the ID payload (e.g. IP address is to be converted from a dotted decimal string into 4 bytes). The property is defined as follows:

NAME	PeerIdentity
DESCRIPTION	Specifies the IKE identity value of the security gateway.
SYNTAX	string

6.16. The Association Class PeerGatewayForTunnel

The class PeerGatewayForTunnel associates IPsecTunnelActions with an ordered list of PeerGateways. The class definition for PeerGatewayForTunnel is as follows:

NAME	PeerGatewayForTunnel
DESCRIPTION	Associates IPsecTunnelActions with an ordered list of PeerGateways.
DERIVED FROM	Dependency (see [CIMCORE])
ABSTRACT	FALSE
PROPERTIES	Antecedent [ref PeerGateway[0..n]] Dependent [ref IPsecTunnelAction[0..n]] SequenceNumber

6.16.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to a PeerGateway instance. The [0..n] cardinality indicates that there an IPsecTunnelAction instance may be associated with zero or more PeerGateway instances.

Note: the cardinality 0 has a specific meaning:

- when the IKE service acts as a responder, this means that the IKE service will accept phase 1 negotiation with any other security gateway;
- when the IKE service acts as an initiator, this means that the IKE service will use the destination IP address (of the IP packets which triggered the SARule) as the IP address of the peer IKE entity.

6.16.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to an IPsecTunnelAction instance. The [0..n] cardinality indicates that a PeerGateway instance may be associated with zero or more IPsecTunnelAction instances.

6.16.3. The Property SequenceNumber

The property `SequenceNumber` specifies the ordering to be used when evaluating `PeerGateway` instances for a given `IPsecTunnelAction`. The property is defined as follows:

NAME	<code>SequenceNumber</code>
DESCRIPTION	Specifies the order of evaluation for <code>PeerGateways</code> .
SYNTAX	unsigned 16-bit integer
VALUE	Lower values are evaluated first.

6.17. The Aggregation Class `ContainedProposal`

The class `ContainedProposal` associates an ordered list of `SAProposals` with the `IKENegotiationAction` that aggregates it. If the referenced `IKENegotiationAction` object is an `IKEAction`, then the referenced `SAProposal` object(s) must be `IKEProposal(s)`. If the referenced `IKENegotiationAction` object is an `IPsecTransportAction` or an `IPsecTunnelAction`, then the referenced `SAProposal` object(s) must be `IPsecProposal(s)`. The class definition for `ContainedProposal` is as follows:

NAME	<code>ContainedProposal</code>
DESCRIPTION	Associates an ordered list of <code>SAProposals</code> with an <code>IKENegotiationAction</code> .
DERIVED FROM	<code>PolicyComponent</code> (see [PCIM])
ABSTRACT	FALSE
PROPERTIES	<code>GroupComponent[ref IKENegotiationAction[0..n]]</code> <code>PartComponent[ref SAProposal[1..n]]</code> <code>SequenceNumber</code>

6.17.1. The Reference `GroupComponent`

- The property `GroupComponent` is inherited from `PolicyComponent` and is overridden to refer to an `IKENegotiationAction` instance. The `[0..n]` cardinality indicates that an `SAProposal` instance may be associated with zero or more `IKENegotiationAction` instances.

6.17.2. The Reference `PartComponent`

The property `PartComponent` is inherited from `PolicyComponent` and is overridden to refer to an `SAProposal` instance. The `[1..n]` cardinality indicates that an `IKENegotiationAction` instance MUST be associated with at least one `SAProposal` instance.

6.17.3. The Property `SequenceNumber`

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

NAME	<code>SequenceNumber</code>
------	-----------------------------

DESCRIPTION Specifies the preference order for the SAProposals.
SYNTAX unsigned 16-bit integer
VALUE Lower-valued proposals are preferred over proposals with higher values. For ContainedProposals that reference the same IKENegotiationAction, SequenceNumber values must be unique.

6.18. The Association Class HostedPeerGatewayInformation

The class HostedPeerGatewayInformation weakly associates a PeerGateway with a System. The class definition for HostedPeerGatewayInformation is as follows:

NAME HostedPeerGatewayInformation
DESCRIPTION Weakly associates a PeerGateway with a System.
DERIVED FROM Dependency (see [[CIMCORE](#)])
ABSTRACT FALSE
PROPERTIES Antecedent [ref System[1..1]]
 Dependent [ref PeerGateway[0..n] [weak]]

6.18.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to a System instance. The [1..1] cardinality indicates that a PeerGateway instance MUST be associated with one and only one System instance.

6.18.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to a PeerGateway instance. The [0..n] cardinality indicates that a System instance may be associated with zero or more PeerGateway instances.

6.19. The Association Class TransformOfPreconfiguredAction

The class TransformOfPreconfiguredAction associates a PreconfiguredSAAction with two, four or six SATransforms that will be applied to the inbound and outbound traffic. The order of application of the SATransforms is implicitly defined in [[IPSEC](#)]. The class definition for TransformOfPreconfiguredAction is as follows:

NAME TransformOfPreconfiguredAction
DESCRIPTION Associates a PreconfiguredSAAction with from one to three SATransforms.
DERIVED FROM Dependency (see [[CIMCORE](#)])
ABSTRACT FALSE
PROPERTIES Antecedent[ref SATransform[2..6]]
 Dependent[ref PreconfiguredSAAction[0..n]]
 SPI
 Direction

6.19.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to an SATransform instance. The [2..6] cardinality indicates that an PreconfiguredSAAction instance may be associated with from two to six SATransform instances.

6.19.2. The Reference Dependent

The property `Dependent` is inherited from `Dependency` and is overridden to refer to a `PreconfiguredSAAction` instance. The `[0..n]` cardinality indicates that an `SATransform` instance may be associated with zero or more `PreconfiguredSAAction` instances.

6.19.3. The Property SPI

The property SPI specifies the SPI to be used by the pre-configured action for the associated transform. The property is defined as follows:

NAME SPI
 DESCRIPTION Specifies the SPI to be used with the SATransform.
 SYNTAX unsigned 32-bit integer

6.19.4. The Property Direction

The property Direction specifies whether the SPI property is for inbound or for outbound traffic. The property is defined as follows:

NAME Direction
 DESCRIPTION Specifies whether the SA is for inbound or outbound traffic.
 SYNTAX unsigned 8-bit integer
 VALUE 1 - this SA is for inbound traffic
 2 - this SA is for outbound traffic

6.20 The Association Class PeerGatewayForPreconfiguredTunnel

The class PeerGatewayForPreconfiguredTunnel associates zero or one PeerGateway with multiple PreconfiguredTunnelActions. The class definition for PeerGatewayForPreconfiguredTunnel is as follows:

NAME PeerGatewayForPreconfiguredTunnel
 DESCRIPTION Associates a PeerGateway with multiple PreconfiguredTunnelAction.
 DERIVED FROM Dependency (see [[CIMCORE](#)])
 ABSTRACT FALSE
 PROPERTIES Antecedent[ref PeerGateway[0..1]]
 Dependent[ref PreconfiguredTunnelAction[0..n]]

6.20.1. The Reference Antecedent

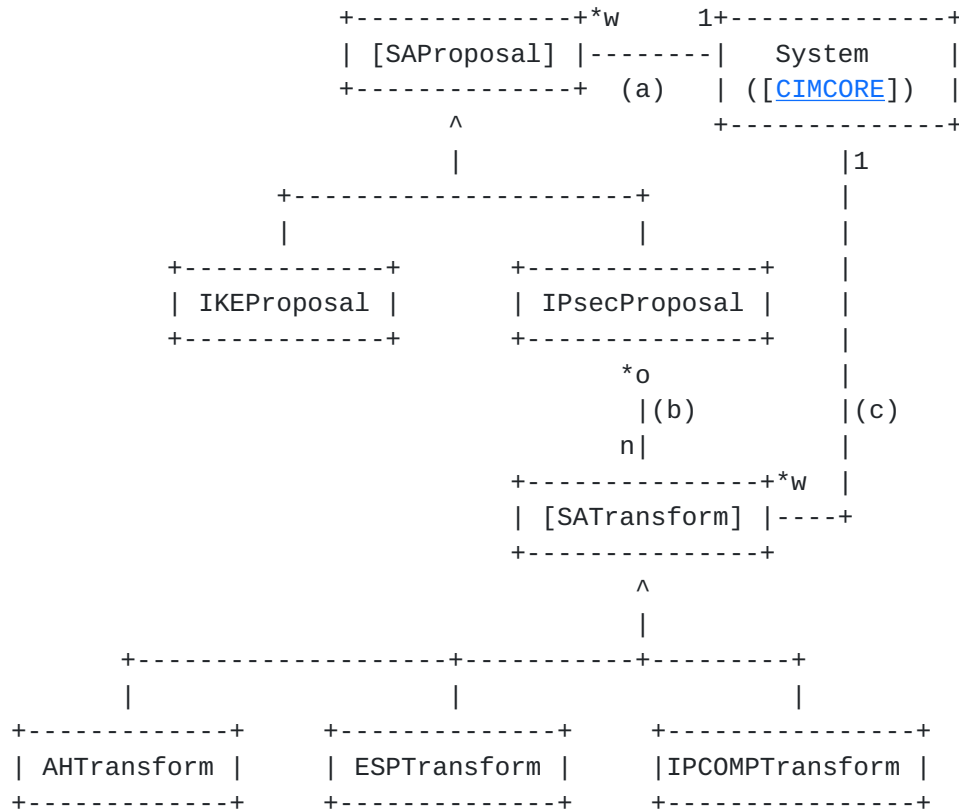
The property Antecedent is inherited from Dependency and is overridden to refer to an PeerGateway instance. The [0..1] cardinality indicates that an PreconfiguredTunnelAction instance may be associated with one PeerGteway instance.

6.20.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to a PreconfiguredTunnelAction instance. The [0..n] cardinality indicates that an PeerGateway instance may be associated with zero or more PreconfiguredSAAction instances.

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) SAPProposalInSystem
- (b) ContainedTransform
- (c) SATransformInSystem

7.1. The Abstract Class SAPProposal

The abstract class SAPProposal 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 SAPProposal is as follows:

```

NAME          SAPProposal
DESCRIPTION    Specifies the common proposal parameters for IKE and
               IPsec security association negotiation.
DERIVED FROM  Policy ([PCIM])
ABSTRACT      TRUE
PROPERTIES    Name
  
```

7.1.1. The Property Name

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

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

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 CipherAlgorithm
 HashAlgorithm
 PRFAlgorithm
 GroupId
 AuthenticationMethod
 MaxLifetimeSeconds
 MaxLifetimeKilobytes
 VendorID

7.2.1. 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 Consult [[IKE](#)] for valid values.

7.2.2. The Property HashAlgorithm

The property HashAlgorithm specifies the proposed phase 1 security association 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 Consult [[IKE](#)] for valid values.

7.2.3. The Property PRFAlgorithm

The property PRFAlgorithm specifies the proposed phase 1 security association pseudo-random function. The property is defined as follows:

NAME PRFAlgorithm
DESCRIPTION Specifies the proposed pseudo-random function for the
 phase 1 security association.
SYNTAX unsigned 16-bit integer
VALUE Currently none defined in [[IKE](#)], if [[IKE](#), [DOI](#)] are
 extended, then the values of [[IKE](#), [DOI](#)] are to be used
 for values of PRFAlgorithm.

7.2.4. The Property GroupId

The property GroupId specifies the proposed phase 1 security association key exchange group. This property is ignored for all aggressive mode exchanges. If the GroupID number is from the vendor-specific range (32768-65535), the property VendorID qualifies the group number. The property is defined as follows:

NAME	GroupId
DESCRIPTION	Specifies the proposed key exchange group for the phase 1 security association.

SYNTAX unsigned 16-bit integer
VALUE Consult [[IKE](#)] for valid values.

Note: the value of this property is to be ignored when doing aggressive mode.

7.2.5. 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 that 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.
 Consult [[IKE](#)] for valid values.

7.2.6. The Property MaxLifetimeSeconds

The property MaxLifetimeSeconds specifies the maximum 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 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.

7.2.7. 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.

7.2.8. The Property VendorID

The property VendorID further qualifies the key exchange group. The property is ignored unless the exchange is not in aggressive mode and the property GroupID is in the vendor-specific range. The property is defined as follows:

NAME	VendorID
------	----------

DESCRIPTION	Specifies the Vendor ID to further qualify the key exchange group.
SYNTAX	string

7.3. The Class IPsecProposal

The class `IPsecProposal` adds no new properties, but inherits `proposal` properties 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 or to be used as a pre-configured action. The class definition for `SATransform` is as follows:

NAME	SATransform
DESCRIPTION	Base class for the different IPsec transforms.
ABSTRACT	TRUE
PROPERTIES	CommonName (from Policy) VendorID MaxLifetimeSeconds MaxLifetimeKilobytes

7.4.1. The Property `CommonName`

The property `CommonName` is inherited from Policy [[PCIM](#)] and specifies a user-friendly name for the SATransform. The property is defined as follows:

NAME	CommonName
DESCRIPTION	Specifies a user-friendly name for this Policy-related object.
SYNTAX	string

7.4.2. 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
a standard one.

7.4.3. The Property MaxLifetimeSeconds

The property MaxLifetimeSeconds specifies the maximum 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 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.

7.4.4. 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.

7.5. 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
UseReplayPrevention
ReplayPreventionWindowSize

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 Consult [[DOI](#)] for valid values.

7.5.2. The Property UseReplayPrevention

The property UseReplayPrevention specifies whether replay prevention detection is to be used. The property is defined as follows:

NAME UseReplayPrevention

DESCRIPTION Specifies whether to enable replay prevention detection.
SYNTAX boolean
VALUE true - replay prevention detection is enabled.
false - replay prevention detection is disabled.

7.5.3. The Property ReplayPreventionWindowSize

The property ReplayPreventionWindowSize specifies, in bits, the length of the sliding window used by the replay prevention detection mechanism. The value of this property is meaningless if UseReplayPrevention is false. It is assumed that the window size will be power of 2. The property is defined as follows:

NAME ReplayPreventionWindowSize
DESCRIPTION Specifies the length of the window used by replay
 prevention detection mechanism.
SYNTAX unsigned 32-bit integer

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
 UseReplayPrevention
 ReplayPreventionWindowSize

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 Consult [[DOI](#)] for valid values.

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 Consult [[DOI](#)] for valid values.

7.6.3. The Property CipherKeyLength

The property CipherKeyLength specifies, in bits, the key length for the ESP encryption algorithm. For encryption algorithms that 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. For encryption algorithms that use fixed number of key rounds, this value is ignored. 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.

7.6.5. The Property UseReplayPrevention

The property UseReplayPrevention specifies whether replay prevention detection is to be used. The property is defined as follows:

NAME	UseReplayPrevention
DESCRIPTION	Specifies whether to enable replay prevention detection.
SYNTAX	boolean
VALUE	true - replay prevention detection is enabled. false - replay prevention detection is disabled.

7.6.6. The Property ReplayPreventionWindowSize

The property ReplayPreventionWindowSize specifies, in bits, the length of the sliding window used by the replay prevention detection mechanism. The value of this property is meaningless if UseReplayPrevention is false. It is assumed that the window size will be power of 2. The property is defined as follows:

NAME	ReplayPreventionWindowSize
DESCRIPTION	Specifies the length of the window used by replay prevention detection mechanism.
SYNTAX	unsigned 32-bit integer

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: a vendor specific algorithm is used and specified in the property PrivateAlgorithm. Consult [DOI] for other valid values.

7.7.2. The Property DictionarySize

The property DictionarySize specifies the log2 maximum size of the dictionary for the compression algorithm. For compression algorithms that have pre-defined dictionary sizes, this value is ignored. 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 Association Class SAProposalInSystem

The class SAProposalInSystem weakly associates SAProposals with a System. The class definition for SAProposalInSystem is as follows:

NAME SAProposalInSystem
DESCRIPTION Weakly associates SAProposals with a System.
DERIVED FROM PolicyInSystem (see [[PCIM](#)])
ABSTRACT FALSE
PROPERTIES Antecedent[ref System [1..1]]
 Dependent[ref SAProposal[0..n] [weak]]

7.8.1. The Reference Antecedent

The property Antecedent is inherited from PolicyInSystem and is overridden to refer to a System instance. The [1..1] cardinality indicates that an SAProposal instance MUST be associated with one and only one System instance.

7.8.2. The Reference Dependent

The property Dependent is inherited from PolicyInSystem and is overridden to refer to an SAProposal instance. The [0..n] cardinality indicates that a System instance may be associated with zero or more SAProposal instances.

7.9. The Aggregation Class ContainedTransform

The class ContainedTransform associates an IPsecProposal with the set

of SATransforms that make up the proposal. If multiple transforms 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 ordered proposal list were

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

then the one sending the proposal would want the other side to pick one from the ESP transform (preferably (HMAC-MD5, 3DES)) list AND one from the AH transform list (preferably MD5).

The class definition for ContainedTransform is as follows:

```
NAME          ContainedTransform
DESCRIPTION    Associates an IPsecProposal with the set of SATransforms
               that make up the proposal.
DERIVED FROM  PolicyComponent (see [PCIM])
ABSTRACT      FALSE
PROPERTIES    GroupComponent[ref IPsecProposal[0..n]]
               PartComponent[ref SATransform[1..n]]
               SequenceNumber
```

7.9.1. The Reference GroupComponent

The property GroupComponent is inherited from PolicyComponent and is overridden to refer to an IPsecProposal instance. The [0..n] cardinality indicates that an SATransform instance may be associated with zero or more IPsecProposal instances.

7.9.2. The Reference PartComponent

The property PartComponent is inherited from PolicyComponent and is overridden to refer to an SATransform instance. The [1..n] cardinality indicates that an IPsecProposal instance MUST be associated with at least one SATransform instance.

7.9.3. 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. For
               ContainedTransforms that reference the same
               IPsecProposal, SequenceNumber values must be unique.
```

7.10. The Association Class SATransformInSystem

The class SATransformInSystem weakly associates SATransforms with a System. The class definition for SATransformInSystem System is as follows:

```
NAME          SATransformInSystem
DESCRIPTION    Weakly associates SATransforms with a System.
DERIVED FROM  PolicyInSystem (see [PCIM])
```

ABSTRACT FALSE
PROPERTIES Antecedent[ref System[1..1]]
 Dependent[ref SATransform[0..n] [weak]]

7.10.1. The Reference Antecedent

The property Antecedent is inherited from PolicyInSystem and is overridden to refer to a System instance. The [1..1] cardinality indicates that an SATransform instance MUST be associated with one and only one System instance.

7.10.2. The Reference Dependent

The property `Dependent` is inherited from `PolicyInSystem` and is overridden to refer to an `SATransform` instance. The `[0..n]` cardinality indicates that a `System` instance may be associated with zero or more `SATransform` instances.

(1) IKEIdentityCredential

This portion of the model contains additional information that is useful in applying the policy. The IKEService class MAY be used to represent the IKE negotiation function in a system. The IKEService uses the various tables that contain information about IKE peers as well as the configuration for specifying security associations that are started automatically. The information in the PeerGateway, PeerIdentityTable and related classes is necessary to completely specify the policies.

An interface (represented by an IPProtocolEndpoint) has an IKEService that provides the negotiation services for that interface. That

service MAY also have a list of security associations automatically started at the time the IKE service is initialized.

The IKEService also has a set of identities that it may use in negotiations with its peers. Those identities are associated with the interfaces (or collections of interfaces).

8.1. The Class IKEService

The class IKEService represents the IKE negotiation function. An instance of this service may provide that negotiation service for one or more interfaces (represented by the IPProtocolEndpoint class) of a System. There may be multiple instances of IKE services on a System but only one per interface. The class definition for IKEService is as follows:

NAME IKEService
DESCRIPTION IKEService is used to represent the IKE negotiation
 function.
DERIVED FROM Service (see [[CIMCORE](#)])
ABSTRACT FALSE

8.2. The Class PeerIdentityTable

The class PeerIdentityTable aggregates the table entries that provide mappings between identities and their addresses. The class definition for PeerIdentityTable is as follows:

NAME PeerIdentityTable
DESCRIPTION PeerIdentityTable aggregates PeerIdentityEntry instances
 to provide a table of identity-address mappings.
DERIVED FROM Collection (see [[CIMCORE](#)])
ABSTRACT FALSE
PROPERTIES Name

8.2.1. The Property Name

The property Name uniquely identifies the table. The property is defined as follows:

NAME Name
DESCRIPTION Name uniquely identifies the table.
SYNTAX string

8.3. The Class PeerIdentityEntry

The class PeerIdentityEntry specifies the mapping between peer identity and their IP address. The class definition for PeerIdentityEntry is as follows:

NAME PeerIdentityEntry
DESCRIPTION PeerIdentityEntry provides a mapping between a peer's
identity and address.
DERIVED FROM LogicalElement (see [[CIMCORE](#)])
ABSTRACT FALSE
PROPERTIES PeerIdentity
PeerIdentityType
PeerAddress
PeerAddressType

The pre-shared key to be used with this peer (if applicable) is
contained in an instance of the class SharedSecret (see [[CIMUSER](#)]).

The pre-shared key is stored in the property Secret, the property protocol contains "IKE", the property algorithm contains the algorithm used to protect the secret (can be "PLAINTEXT" if the IPsec entity has no secret storage), the value of property RemoteID must match the PeerIdentity property of the PeerIdentityEntry instance describing the IKE peer.

8.3.1. The Property PeerIdentity

The property PeerIdentity contains a string encoding of the Identity payload for the IKE peer. The property is defined as follows:

NAME	PeerIdentity
DESCRIPTION	The PeerIdentity is the ID payload of a peer.
SYNTAX	string

8.3.2. The Property PeerIdentityType

The property PeerIdentityType is an enumeration that specifies the type of the PeerIdentity. The property is defined as follows:

NAME	PeerIdentityType
DESCRIPTION	PeerIdentityType is the type of the ID payload of a peer.
SYNTAX	unsigned 16-bit integer
VALUE	The enumeration values are specified in [DOI] section 4.6.2.1 .

8.3.3. The Property PeerAddress

The property PeerAddress specifies the string representation of the IP address of the peer formatted according to the appropriate convention as defined in the PeerAddressType property (e.g., dotted decimal notation). The property is defined as follows:

NAME	PeerAddress
DESCRIPTION	PeerAddress is the address of the peer with the ID payload.
SYNTAX	string
VALUE	String representation of an IPv4 or IPv6 address.

8.3.4. The Property PeerAddressType

The property PeerAddressType specifies the format of the PeerAddress property value. The property is defined as follows:

NAME	PeerAddressType
DESCRIPTION	PeerAddressType is the type of address in PeerAddress.
SYNTAX	unsigned 16-bit integer
VALUE	0 - Unknown

- 1 - IPv4
- 2 - IPv6

8.4. The Class AutostartIKEConfiguration

The class AutostartIKEConfiguration groups AutostartIKESetting instances into configuration sets. When applied, the settings cause an IKE service to automatically start (negotiate or statically set as appropriate) the Security Associations. The class definition for AutostartIKEConfiguration is as follows:

NAME AutostartIKEConfiguration

DESCRIPTION A configuration set of AutostartIKESetting instances to be automatically started by the IKE service.

DERIVED FROM SystemConfiguration (see [[CIMCORE](#)])

ABSTRACT FALSE

8.5. The Class AutostartIKESetting

The class AutostartIKESetting is used to automatically initiate IKE negotiations with peers (or statically create an SA) as specified in the AutostartIKESetting properties. Appropriate actions are initiated according to the policy that matches the setting parameters. The class definition for AutostartIKESetting is as follows:

NAME AutostartIKESetting

DESCRIPTION AutostartIKESetting is used to automatically initiate IKE negotiations with peers or statically create an SA.

DERIVED FROM SystemSetting (see [[CIMCORE](#)])

ABSTRACT FALSE

PROPERTIES Phase1Only
AddressType
SourceAddress
SourcePort
DestinationAddress
DestinationPort
Protocol

8.5.1. The Property Phase1Only

The property Phase1Only is used to limit the IKE negotiation to a phase 1 SA establishment only. When set to False, both phase 1 and phase 2 SAs are negotiated.

The property is defined as follows:

NAME Phase1Only

DESCRIPTION Used to indicate which security associations to attempt to establish (phase 1 only, or phase 1 and 2).

SYNTAX boolean

VALUE true - attempt to establish a phase 1 security association
false - attempt to establish phase 1 and phase 2 security associations

8.5.2. The Property AddressType

The property AddressType specifies type of the addresses in the SourceAddress and DestinationAddress properties. The property is defined as follows:

NAME AddressType

DESCRIPTION AddressType is the type of address in SourceAddress and DestinationAddress properties.

SYNTAX unsigned 16-bit integer

VALUE 0 - Unknown
1 - IPv4
2 - IPv6

8.5.3. The Property SourceAddress

The property SourceAddress specifies the dotted-decimal or colon-decimal formatted IP address used as the source address in comparing

with policy filter entries and used in any phase 2 negotiations. The property is defined as follows:

NAME	SourceAddress
DESCRIPTION	The source address to compare with the filters to determine the appropriate policy rule.
SYNTAX	string
VALUE	dotted-decimal or colon-decimal formatted IP address

8.5.4. The Property SourcePort

The property SourcePort specifies the port number used as the source port in comparing with policy filter entries and used in any phase 2 negotiations. The property is defined as follows:

NAME	SourcePort
DESCRIPTION	The source port to compare with the filters to determine the appropriate policy rule.
SYNTAX	unsigned 16-bit integer

8.5.5. The Property DestinationAddress

The property DestinationAddress specifies the dotted-decimal or colon-decimal formatted IP address used as the destination address in comparing with policy filter entries and used in any phase 2 negotiations. The property is defined as follows:

NAME	DestinationAddress
DESCRIPTION	The destination address to compare with the filters to determine the appropriate policy rule.
SYNTAX	string
VALUE	dotted-decimal or colon-decimal formatted IP address

8.5.6. The Property DestinationPort

The property DestinationPort specifies the port number used as the destination port in comparing with policy filter entries and used in any phase 2 negotiations. The property is defined as follows:

NAME	DestinationPort
DESCRIPTION	The destination port to compare with the filters to determine the appropriate policy rule.
SYNTAX	unsigned 16-bit integer

8.5.7. The Property Protocol

The property Protocol specifies the protocol number used in comparing with policy filter entries and used in any phase 2 negotiations. The property is defined as follows:

NAME	Protocol
DESCRIPTION	The protocol number used in comparing with policy filter entries.
SYNTAX	unsigned 8-bit integer

8.6. The Class IKEIdentity

The class IKEIdentity is used to represent the identities that may be used for an IPProtocolEndpoint (or collection of IPProtocolEndpoints) to identify the IKE Service in IKE phase 1 negotiations. The policy IKEAction.UseIKEIdentityType specifies which type of the available identities to use in a negotiation exchange and the

IKERule.IdentityContexts specifies the match values to be used, along with the local address, in selecting the appropriate identity for a negotiation. The ElementID property value (defined in the parent class, UsersAccess) should be that of either the IPProtocolEndpoint or Collection of endpoints as appropriate. The class definition for IKEIdentity is as follows:

NAME	IKEIdentity
DESCRIPTION	IKEIdentity is used to represent the identities that may be used for an IPProtocolEndpoint (or collection of IPProtocolEndpoints) to identify the IKE Service in IKE phase 1 negotiations.
DERIVED FROM	UsersAccess (see [CIMUSER])
ABSTRACT	FALSE
PROPERTIES	IdentityType IdentityValue IdentityContexts

8.6.1. The Property IdentityType

The property IdentityType is an enumeration that specifies the type of the IdentityValue. The property is defined as follows:

NAME	IdentityType
DESCRIPTION	IdentityType is the type of the IdentityValue.
SYNTAX	unsigned 16-bit integer
VALUE	The enumeration values are specified in [DOI] section 4.6.2.1 .

8.6.2. The Property IdentityValue

The property IdentityValue contains a string encoding of the Identity payload. For IKEIdentity instances that are address types (i.e. IPv4 or IPv6 addresses), the IdentityValue string value MAY be omitted; then the associated IPProtocolEndpoint (or appropriate member of the Collection of endpoints) is used as the identity value. The property is defined as follows:

NAME	IdentityValue
DESCRIPTION	IdentityValue contains a string encoding of the Identity payload.
SYNTAX	string

8.6.3. The Property IdentityContexts

The IdentityContexts property is used to constrain the use of IKEIdentity instances to match that specified in the IKERule.IdentityContexts. The IdentityContexts are formatted as policy roles and role combinations [[PCIM](#)] & [[PCIMe](#)]. Each value represents one context or context combination. Since this is a

multi-valued property, more than one context or combination of contexts can be associated with a single IKEIdentity. Each value is a string of the form: `<ContextName>[&&<ContextName>]*` where the individual context names appear in alphabetical order (according to the collating sequence for UCS-2). If one or more values in the `IKERule.IdentityContexts` array match one or more `IKEIdentity.IdentityContexts` then the identity's context matches. (That is, each value of the `IdentityContext` array is an ORed condition.) In combination with the address of the `IPProtocolEndpoint` and `IKEAction.UseIKEIdentityType`, there SHOULD be exactly one `IKEIdentity`. The property is defined as follows:

NAME IdentityContexts
DESCRIPTION The IKE service of a security endpoint may have multiple identities for use in different situations. The combination of the interface (represented by the IPProtocolEndpoint), the identity type (as specified in the IKEAction) and the IdentityContexts selects a unique identity.
SYNTAX string array
VALUE string of the form <ContextName>[&&<ContextName>]*

8.7. The Association Class HostedPeerIdentityTable

The class HostedPeerIdentityTable provides the name scoping relationship for PeerIdentityTable entries in a System. The PeerIdentityTable is weak to the System. The class definition for HostedPeerIdentityTable is as follows:

NAME HostedPeerIdentityTable
DESCRIPTION The PeerIdentityTable instances are weak (name scoped by) the owning System.
DERIVED FROM Dependency (see [[CIMCORE](#)])
ABSTRACT FALSE
PROPERTIES Antecedent [ref System[1..1]]
Dependent [ref PeerIdentityTable[0..n] [weak]]

8.7.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to a System instance. The [1..1] cardinality indicates that a PeerIdentityTable instance MUST be associated in a weak relationship with one and only one System instance.

8.7.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to a PeerIdentityTable instance. The [0..n] cardinality indicates that a System instance may be associated with zero or more PeerIdentityTable instances.

8.8. The Aggregation Class PeerIdentityMember

The class PeerIdentityMember aggregates PeerIdentityEntry instances into a PeerIdentityTable. This is a weak aggregation. The class definition for PeerIdentityMember is as follows:

NAME PeerIdentityMember
DESCRIPTION PeerIdentityMember aggregates PeerIdentityEntry instances into a PeerIdentityTable.
DERIVED FROM MemberOfCollection (see [[CIMCORE](#)])
ABSTRACT FALSE

```
PROPERTIES  Collection [ref PeerIdentityTable[1..1]]
            Member [ref PeerIdentityEntry [0..n] [weak]]
```

8.8.1. The Reference Collection

The property Collection is inherited from MemberOfCollection and is overridden to refer to a PeerIdentityTable instance. The [1..1] cardinality indicates that a PeerIdentityEntry instance MUST be associated with one and only one PeerIdentityTable instance (i.e., PeerIdentityEntry instances are not shared across PeerIdentityTables).

8.8.2. The Reference Member

The property Member is inherited from MemberOfCollection and is overridden to refer to a PeerIdentityEntry instance. The [0..n] cardinality indicates that a PeerIdentityTable instance may be associated with zero or more PeerIdentityEntry instances.

8.9. The Association Class IKEServicePeerGateway

The class IKEServicePeerGateway provides the association between an IKEService and the list of PeerGateway instances that it uses in negotiating with security gateways. The class definition for IKEServicePeerGateway is as follows:

NAME	IKEServicePeerGateway
DESCRIPTION	Associates an IKEService and the list of PeerGateway instances that it uses in negotiating with security gateways.
DERIVED FROM	Dependency (see [CIMCORE])
ABSTRACT	FALSE
PROPERTIES	Antecedent [ref PeerGateway[0..n]] Dependent [ref IKEService[0..n]]

8.9.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to a PeerGateway instance. The [0..n] cardinality indicates that an IKEService instance may be associated with zero or more PeerGateway instances.

8.9.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to an IKEService instance. The [0..n] cardinality indicates that a PeerGateway instance may be associated with zero or more IKEService instances.

8.10. The Association Class IKEServicePeerIdentityTable

The class IKEServicePeerIdentityTable provides the relationship between an IKEService and a PeerIdentityTable that it uses to map between addresses and identities as required. The class definition for IKEServicePeerIdentityTable is as follows:

NAME	IKEServicePeerIdentityTable
DESCRIPTION	IKEServicePeerIdentityTable provides the relationship between an IKEService and a PeerIdentityTable that it uses.
DERIVED FROM	Dependency (see [CIMCORE])
ABSTRACT	FALSE

```
PROPERTIES  Antecedent [ref PeerIdentityTable[0..n]]
            Dependent  [ref IKEService[0..n]]
```

8.10.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to a PeerIdentityTable instance. The [0..n] cardinality indicates that an IKEService instance may be associated with zero or more PeerIdentityTable instances.

8.10.2. The Reference Dependent

The property `Dependent` is inherited from `Dependency` and is overridden to refer to an `IKEService` instance. The `[0..n]` cardinality indicates that a `PeerIdentityTable` instance may be associated with zero or more `IKEService` instances.

8.11. The Association Class `IKEAutostartSetting`

The class `IKEAutostartSetting` associates an `AutostartIKESetting` with an `IKEService` that may use it to automatically start an IKE negotiation or create a static SA. The class definition for `IKEAutostartSetting` is as follows:

```
NAME           IKEAutostartSetting
DESCRIPTION    Associates a AutostartIKESetting with an IKEService.
DERIVED FROM   ElementSetting (see [CIMCORE])
ABSTRACT      FALSE
PROPERTIES     Element [ref IKEService[0..n]]
                Setting [ref AutostartIKESetting[0..n]]
```

8.11.1. The Reference Element

The property `Element` is inherited from `ElementSetting` and is overridden to refer to an `IKEService` instance. The `[0..n]` cardinality indicates an `AutostartIKESetting` instance may be associated with zero or more `IKEService` instances.

8.11.2. The Reference Setting

The property `Setting` is inherited from `ElementSetting` and is overridden to refer to an `AutostartIKESetting` instance. The `[0..n]` cardinality indicates that an `IKEService` instance may be associated with zero or more `AutostartIKESetting` instances.

8.12. The Aggregation Class `AutostartIKESettingContext`

The class `AutostartIKESettingContext` aggregates the settings used to automatically start negotiations or create a static SA into a configuration set. The class definition for `AutostartIKESettingContext` is as follows:

```
NAME           AutostartIKESettingContext
DESCRIPTION    AutostartIKESettingContext aggregates the
                AutostartIKESetting instances into a configuration set.
DERIVED FROM   SystemSettingContext (see [CIMCORE])
ABSTRACT      FALSE
PROPERTIES     Context [ref AutostartIKEConfiguration [0..n]]
                Setting [ref AutostartIKESetting [0..n]]
                SequenceNumber
```

8.12.1. The Reference Context

The property Context is inherited from SystemSettingContext and is overridden to refer to an AutostartIKEConfiguration instance. The [0..n] cardinality indicates that an AutostartIKESetting instance may be associated with zero or more AutostartIKEConfiguration instances (i.e., a setting may be in multiple configuration sets).

8.12.2. The Reference Setting

The property Setting is inherited from SystemSettingContext and is overridden to refer to an AutostartIKESetting instance. The [0..n]

cardinality indicates that an AutostartIKEConfiguration instance may be associated with zero or more AutostartIKESetting instances.

8.12.3. The Property SequenceNumber

The property SequenceNumber specifies indicates the ordering to be used when starting negotiations or creating a static SA. A zero value indicates that order is not significant and settings may be applied in parallel with other settings. All other settings in the configuration are executed in sequence from lower values to high. Sequence numbers need not be unique in an AutostartIKEConfiguration and order is not significant for settings with the same sequence number. The property is defined as follows:

NAME	SequenceNumber
DESCRIPTION	The sequence in which the settings are applied within a configuration set.
SYNTAX	unsigned 16-bit integer

8.13. The Association Class IKEServiceForEndpoint

The class IKEServiceForEndpoint provides the association showing which IKE service, if any, provides IKE negotiation services for which network interfaces. The class definition for IKEServiceForEndpoint is as follows:

NAME	IKEServiceForEndpoint
DESCRIPTION	Associates an IPProtocolEndpoint with an IKEService that provides negotiation services for the endpoint.
DERIVED FROM	Dependency (see [CIMCORE])
ABSTRACT	FALSE
PROPERTIES	Antecedent [ref IKEService[0..1]] Dependent [ref IPProtocolEndpoint[0..n]]

8.13.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to an IKEService instance. The [0..1] cardinality indicates that an IPProtocolEndpoint instance MUST be associated with at most one IKEService instance.

8.13.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to an IPProtocolEndpoint that is associated with at most one IKEService. The [0..n] cardinality indicates an IKEService instance may be associated with zero or more IPProtocolEndpoint instances.

8.14. The Association Class IKEAutostartConfiguration

The class IKEAutostartConfiguration provides the relationship between an IKEService and a configuration set that it uses to automatically start a set of SAs. The class definition for IKEAutostartConfiguration is as follows:

NAME	IKEAutostartConfiguration
DESCRIPTION	IKEAutostartConfiguration provides the relationship between an IKEService and an AutostartIKEConfiguration that it uses to automatically start a set of SAs.
DERIVED FROM	Dependency (see [CIMCORE])
ABSTRACT	FALSE

PROPERTIES Antecedent [ref AutostartIKEConfiguration [0..n]]
 Dependent [ref IKEService [0..n]]
 Active

8.14.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to an AutostartIKEConfiguration instance. The [0..n] cardinality indicates that an IKEService instance may be associated with zero or more AutostartIKEConfiguration instances.

8.14.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to an IKEService instance. The [0..n] cardinality indicates that an AutostartIKEConfiguration instance may be associated with zero or more IKEService instances.

8.14.3. The Property Active

The property Active specifies indicates whether the AutostartIKEConfiguration set is currently active for the associated IKEService. That is, at boot time, the active configuration is used to automatically start IKE negotiations and create static SAs. The property is defined as follows:

NAME Active
 DESCRIPTION Active indicates whether the AutostartIKEConfiguration set is currently active for the associated IKEService.
 SYNTAX boolean
 VALUE true - AutostartIKEConfiguration is currently active for associated IKEService.
 false - AutostartIKEConfiguration is currently inactive for associated IKEService.

8.15. The Association Class IKEUsesCredentialManagementService

The class IKEUsesCredentialManagementService defines the set of CredentialManagementService(s) that are trusted sources of credentials for IKE phase 1 negotiations. The class definition for IKEUsesCredentialManagementService is as follows:

NAME IKEUsesCredentialManagementService
 DESCRIPTION Associates the set of CredentialManagementService(s) that are trusted by the IKEService as sources of credentials used in IKE phase 1 negotiations.
 DERIVED FROM Dependency (see [[CIMCORE](#)])
 ABSTRACT FALSE
 PROPERTIES Antecedent [ref CredentialManagementService [0..n]]
 Dependent [ref IKEService [0..n]]

8.15.1. The Reference Antecedent

The property Antecedent is inherited from Dependency and is overridden to refer to a CredentialManagementService instance. The [0..n] cardinality indicates that an IKEService instance may be associated with zero or more CredentialManagementService instances.

8.15.2. The Reference Dependent

The property Dependent is inherited from Dependency and is overridden to refer to an IKEService instance. The [0..n] cardinality indicates

that a `CredentialManagementService` instance may be associated with zero or more `IKEService` instances.

8.16. The Association Class `EndpointHasLocalIKEIdentity`

The class `EndpointHasLocalIKEIdentity` associates an `IPProtocolEndpoint` with a set of `IKEIdentity` instances that may be used in negotiating security associations on the endpoint. An `IKEIdentity` MUST be associated with either an `IPProtocolEndpoint` using this association or with a collection of `IKEIdentity` instances using the `CollectionHasLocalIKEIdentity` association. The class definition for `EndpointHasLocalIKEIdentity` is as follows:

NAME `EndpointHasLocalIKEIdentity`
DESCRIPTION `EndpointHasLocalIKEIdentity` associates an
 `IPProtocolEndpoint` with a set of `IKEIdentity` instances.
DERIVED FROM `ElementAsUser` (see [[CIMUSER](#)])
ABSTRACT FALSE
PROPERTIES Antecedent [`ref IPProtocolEndpoint` [0..1]]
 Dependent [`ref IKEIdentity` [0..n]]

8.16.1. The Reference Antecedent

The property `Antecedent` is inherited from `ElementAsUser` and is overridden to refer to an `IPProtocolEndpoint` instance. The [0..1] cardinality indicates that an `IKEIdentity` instance MUST be associated with at most one `IPProtocolEndpoint` instance.

8.16.2. The Reference Dependent

The property `Dependent` is inherited from `ElementAsUser` and is overridden to refer to an `IKEIdentity` instance. The [0..n] cardinality indicates that an `IPProtocolEndpoint` instance may be associated with zero or more `IKEIdentity` instances.

8.17. The Association Class `CollectionHasLocalIKEIdentity`

The class `CollectionHasLocalIKEIdentity` associates a `Collection` of `IPProtocolEndpoint` instances with a set of `IKEIdentity` instances that may be used in negotiating SAs for endpoints in the collection. An `IKEIdentity` MUST be associated with either an `IPProtocolEndpoint` using the `EndpointHasLocalIKEIdentity` association or with a collection of `IKEIdentity` instances using this association. The class definition for `CollectionHasLocalIKEIdentity` is as follows:

NAME `CollectionHasLocalIKEIdentity`
DESCRIPTION `CollectionHasLocalIKEIdentity` associates a collection of
 `IPProtocolEndpoint` instances with a set of `IKEIdentity`
 instances.
DERIVED FROM `ElementAsUser` (see [[CIMUSER](#)])

ABSTRACT FALSE
PROPERTIES Antecedent [ref Collection [0..1]]
Dependent [ref IKEIdentity [0..n]]

8.17.1. The Reference Antecedent

The property Antecedent is inherited from ElementAsUser and is overridden to refer to a Collection instance. The [0..1] cardinality indicates that an IKEIdentity instance MUST be associated with at most one Collection instance.

8.17.2. The Reference Dependent

The property `Dependent` is inherited from `ElementAsUser` and is overridden to refer to an `IKEIdentity` instance. The `[0..n]` cardinality indicates that a `Collection` instance may be associated with zero or more `IKEIdentity` instances.

8.18. The Association Class `IKEIdentitysCredential`

The class `IKEIdentitysCredential` is an association that relates a set of credentials to their corresponding local IKE Identities. The class definition for `IKEIdentitysCredential` is as follows:

```
NAME           IKEIdentitysCredential
DESCRIPTION    IKEIdentitysCredential associates a set of credentials
               to their corresponding local IKEIdentity.
DERIVED FROM   UsersCredential (see [CIMCORE])
ABSTRACT       FALSE
PROPERTIES     Antecedent [ref Credential [0..n]]
               Dependent [ref IKEIdentity [0..n]]
```

8.18.1. The Reference Antecedent

The property `Antecedent` is inherited from `UsersCredential` and is overridden to refer to a `Credential` instance. The `[0..n]` cardinality indicates that `IKEIdentity` instance may be associated with zero or more `Credential` instances.

8.18.2. The Reference Dependent

The property `Dependent` is inherited from `UsersCredential` and is overridden to refer to an `IKEIdentity` instance. The `[0..n]` cardinality indicates that a `Credential` instance may be associated with zero or more `IKEIdentity` instances.

9. Implementation Requirements

The following table specifies which classes, properties, associations and aggregations **MUST** or **SHOULD** or **MAY** be implemented.

4. Policy Classes	
4.1. The Class <code>IPsecPolicyGroup</code>	MUST
4.2. The Class <code>SARule</code>	MUST
4.2.1. The Property <code>PolicyRuleName</code>	MAY
4.2.1. The Property <code>Enabled</code>	MUST
4.2.1. The Property <code>ConditionListType</code>	MUST
4.2.1. The Property <code>RuleUsage</code>	MAY
4.2.1. The Property <code>Mandatory</code>	MAY
4.2.1. The Property <code>SequencedActions</code>	MUST
4.2.1. The Property <code>PolicyRoles</code>	MAY
4.2.1. The Property <code>PolicyDecisionStrategy</code>	MAY

4.2.2	The Property ExecutionStrategy.....	MUST
4.2.3	The Property LimitNegotiation.....	MAY
4.3.	The Class IKERule.....	MUST
4.3.1.	The Property IdentityContexts.....	MAY
4.4.	The Class IPsecRule.....	MUST
4.5.	The Association Class IPsecPolicyForEndpoint.....	MAY
4.5.1.	The Reference Antecedent.....	MUST
4.5.2.	The Reference Dependent.....	MUST
4.6.	The Association Class IPsecPolicyForSystem.....	MAY
4.6.1.	The Reference Antecedent.....	MUST
4.6.2.	The Reference Dependent.....	MUST
4.7.	The Aggregation Class SARuleInPolicyGroup.....	MUST

4.7.1. The Property Priority.....	SHOULD
4.7.2. The Reference GroupComponent.....	MUST
4.7.3. The Reference PartComponent.....	MUST
4.8. The Aggregation Class SAConditionInRule.....	MUST
4.8.1. The Property GroupNumber.....	SHOULD
4.8.1. The Property ConditionNegated.....	SHOULD
4.8.2. The Reference GroupComponent.....	MUST
4.8.3. The Reference PartComponent.....	MUST
4.9. The Aggregation Class PolicyActionInSARule.....	MUST
4.9.1. The Reference GroupComponent.....	MUST
4.9.2. The Reference PartComponent.....	MUST
4.9.3. The Property ActionOrder.....	SHOULD
5. Condition and Filter Classes	
5.1. The Class SACondition.....	MUST
5.2. The Class IPHeadersFilter.....	SHOULD
5.3. The Class CredentialFilterEntry.....	MAY
5.3.1. The Property MatchFieldName.....	MUST
5.3.2. The Property MatchFieldValue.....	MUST
5.3.3. The Property CredentialType.....	MUST
5.4. The Class IPSOFilterEntry.....	MAY
5.4.1. The Property MatchConditionType.....	MUST
5.4.2. The Property MatchConditionValue.....	MUST
5.5. The Class PeerIDPayloadFilterEntry.....	MAY
5.5.1. The Property MatchIdentityType.....	MUST
5.5.2. The Property MatchIdentityValue.....	MUST
5.6. The Association Class FilterOfSACondition.....	SHOULD
5.6.1. The Reference Antecedent.....	MUST
5.6.2. The Reference Dependent.....	MUST
5.7. The Association Class AcceptCredentialFrom.....	MAY
5.7.1. The Reference Antecedent.....	MUST
5.7.2. The Reference Dependent.....	MUST
6. Action Classes	
6.1. The Class SAAction.....	MUST
6.1.1. The Property DoActionLogging.....	MAY
6.1.2. The Property DoPacketLogging.....	MAY
6.2. The Class SASStaticAction.....	MUST
6.2.1. The Property LifetimeSeconds.....	MUST
6.3. The Class IPsecBypassAction.....	SHOULD
6.4. The Class IPsecDiscardAction.....	SHOULD
6.5. The Class IKERejectAction.....	MAY
6.6. The Class PreconfiguredSAAction.....	MUST
6.6.1. The Property LifetimeKilobytes.....	MUST
6.7. The Class PreconfiguredTransportAction.....	MUST
6.8. The Class PreconfiguredTunnelAction.....	MUST
6.8.1. The Property DFHandling.....	MUST
6.9. The Class SANegotiationAction.....	MUST
6.10. The Class IKENegotiationAction.....	MUST
6.10.1. The Property MinLifetimeSeconds.....	MAY
6.10.2. The Property MinLifetimeKilobytes.....	MAY

6.10.3. The Property IdleDurationSeconds.....MAY
6.11. The Class IPsecAction.....MUST
6.11.1. The Property UsePFS.....MUST
6.11.2. The Property UseIKEGroup.....MAY
6.11.3. The Property GroupId.....MUST
6.11.4. The Property Granularity.....SHOULD
6.11.5. The Property VendorID.....MAY
6.12. The Class IPsecTransportAction.....MUST
6.13. The Class IPsecTunnelAction.....MUST
6.13.1. The Property DFHandling.....MUST
6.14. The Class IKEAction.....MUST
6.14.1. The Property ExchangeModeMUST

- 6.14.2. The Property UseIKEIdentityType.....MUST
- 6.14.3. The Property VendorID.....MAY
- 6.14.4. The Property AggressiveModeGroupId.....MAY
- 6.15. The Class PeerGateway.....MUST
 - 6.15.1. The Property Name.....SHOULD
 - 6.15.2. The Property PeerIdentityType.....MUST
 - 6.15.3. The Property PeerIdentity.....MUST
- 6.16. The Association Class PeerGatewayForTunnel.....MUST
 - 6.16.1. The Reference Antecedent.....MUST
 - 6.16.2. The Reference Dependent.....MUST
 - 6.16.3. The Property SequenceNumber.....SHOULD
- 6.17. The Aggregation Class ContainedProposal.....MUST
 - 6.17.1. The Reference GroupComponent.....MUST
 - 6.17.2. The Reference PartComponent.....MUST
 - 6.17.3. The Property SequenceNumber.....MUST
- 6.18. The Association Class HostedPeerGatewayInformation.....MAY
 - 6.18.1. The Reference Antecedent.....MUST
 - 6.18.2. The Reference Dependent.....MUST
- 6.19. The Association Class TransformOfPreconfiguredAction....MUST
 - 6.19.1. The Reference Antecedent.....MUST
 - 6.19.2. The Reference Dependent.....MUST
 - 6.19.3. The Property SPI.....MUST
 - 6.19.4. The Property Direction.....MUST
- 6.20. The Association Class PeerGatewayForPreconfiguredTunnel..MUST
 - 6.20.1. The Reference Antecedent.....MUST
 - 6.20.2. The Reference Dependent.....MUST
- 7. Proposal and Transform Classes
 - 7.1. The Abstract Class SAProposal.....MUST
 - 7.1.1. The Property Name.....SHOULD
 - 7.2. The Class IKEProposal.....MUST
 - 7.2.1. The Property CipherAlgorithm.....MUST
 - 7.2.2. The Property HashAlgorithm.....MUST
 - 7.2.3. The Property PRFAlgorithm.....MAY
 - 7.2.4. The Property GroupId.....MUST
 - 7.2.5. The Property AuthenticationMethod.....MUST
 - 7.2.6. The Property MaxLifetimeSeconds.....MUST
 - 7.2.7. The Property MaxLifetimeKilobytes.....MUST
 - 7.2.8. The Property VendorID.....MAY
 - 7.3. The Class IPsecProposal.....MUST
 - 7.4. The Abstract Class SATransform.....MUST
 - 7.4.1. The Property TransformName.....SHOULD
 - 7.4.2. The Property VendorID.....MAY
 - 7.4.3. The Property MaxLifetimeSeconds.....MUST
 - 7.4.4. The Property MaxLifetimeKilobytes.....MUST
 - 7.5. The Class AHTransform.....MUST
 - 7.5.1. The Property AHTransformId.....MUST
 - 7.5.2. The Property UseReplayPrevention.....MAY
 - 7.5.3. The Property ReplayPreventionWindowSize.....MAY
 - 7.6. The Class ESPTransform.....MUST

- 7.6.1. The Property IntegrityTransformId.....MUST
- 7.6.2. The Property CipherTransformId.....MUST
- 7.6.3. The Property CipherKeyLength.....MAY
- 7.6.4. The Property CipherKeyRounds.....MAY
- 7.6.5. The Property UseReplayPrevention.....MAY
- 7.6.6. The Property ReplayPreventionWindowSize.....MAY
- 7.7. The Class IPCOMPTransform.....MAY
- 7.7.1. The Property Algorithm.....MUST
- 7.7.2. The Property DictionarySize.....MAY
- 7.7.3. The Property PrivateAlgorithm.....MAY
- 7.8. The Association Class SAProposalInSystem.....MAY
- 7.8.1. The Reference Antecedent.....MUST
- 7.8.2. The Reference Dependent.....MUST

7.9. The Aggregation Class ContainedTransform.....	MUST
7.9.1. The Reference GroupComponent.....	MUST
7.9.2. The Reference PartComponent.....	MUST
7.9.3. The Property SequenceNumber.....	MUST
7.10. The Association Class SATransformInSystem.....	MAY
7.10.1. The Reference Antecedent.....	MUST
7.10.2. The Reference Dependent.....	MUST
8. IKE Service and Identity Classes	
8.1. The Class IKEService.....	MAY
8.2. The Class PeerIdentityTable.....	MAY
8.3.1. The Property Name.....	SHOULD
8.3. The Class PeerIdentityEntry.....	MAY
8.3.1. The Property PeerIdentity.....	SHOULD
8.3.2. The Property PeerIdentityType.....	SHOULD
8.3.3. The Property PeerAddress.....	SHOULD
8.3.4. The Property PeerAddressType.....	SHOULD
8.4. The Class AutostartIKEConfiguration.....	MAY
8.5. The Class AutostartIKESetting.....	MAY
8.5.1. The Property Phase1Only.....	MAY
8.5.2. The Property AddressType.....	SHOULD
8.5.3. The Property SourceAddress.....	MUST
8.5.4. The Property SourcePort.....	MUST
8.5.5. The Property DestinationAddress.....	MUST
8.5.6. The Property DestinationPort.....	MUST
8.5.7. The Property Protocol.....	MUST
8.6. The Class IKEIdentity.....	MAY
8.6.1. The Property IdentityType.....	MUST
8.6.2. The Property IdentityValue.....	MUST
8.6.3. The Property IdentityContexts.....	MAY
8.7. The Association Class HostedPeerIdentityTable.....	MAY
8.7.1. The Reference Antecedent.....	MUST
8.7.2. The Reference Dependent.....	MUST
8.8. The Aggregation Class PeerIdentityMember.....	MAY
8.8.1. The Reference Collection.....	MUST
8.8.2. The Reference Member.....	MUST
8.9. The Association Class IKEServicePeerGateway.....	MAY
8.9.1. The Reference Antecedent.....	MUST
8.9.2. The Reference Dependent.....	MUST
8.10. The Association Class IKEServicePeerIdentityTable.....	MAY
8.10.1. The Reference Antecedent.....	MUST
8.10.2. The Reference Dependent.....	MUST
8.11. The Association Class IKEAutostartSetting.....	MAY
8.11.1. The Reference Element.....	MUST
8.11.2. The Reference Setting.....	MUST
8.12. The Aggregation Class AutostartIKESettingContext.....	MAY
8.12.1. The Reference Context.....	MUST
8.12.2. The Reference Setting.....	MUST
8.12.3. The Property SequenceNumber.....	SHOULD
8.13. The Association Class IKEServiceForEndpoint.....	MAY

8.13.1. The Reference Antecedent.....MUST
8.13.2. The Reference Dependent.....MUST
8.14. The Association Class IKEAutostartConfiguration.....MAY
8.14.1. The Reference Antecedent.....MUST
8.14.2. The Reference Dependent.....MUST
8.14.3. The Property Active.....SHOULD
8.15. The Association Class IKEUsesCredentialManagementService..MAY
8.15.1. The Reference Antecedent.....MUST
8.15.2. The Reference Dependent.....MUST
8.16. The Association Class EndpointHasLocalIKEIdentity.....MAY
8.16.1. The Reference Antecedent.....MUST
8.16.2. The Reference Dependent.....MUST
8.17. The Association Class CollectionHasLocalIKEIdentity.....MAY

8.17.1. The Reference Antecedent.....MUST
8.17.2. The Reference Dependent.....MUST
8.18. The Association Class IKEIdentityCredential.....MAY
8.18.1. The Reference Antecedent.....MUST
8.18.2. The Reference Dependent.....MUST

10. 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.

11. 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.

12. Acknowledgments

The authors would like to thank Mike Jeronimo, Ylian Saint-Hilaire, Vic Lortz, William Dixon, Man Li, Wes Hardaker and Ricky Charlet 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.

13. 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. Ellessen, J. Strassner, "Policy Core Information Model -- Version 1 Specification", [RFC 3060](#), February 2001.

[PCIME] Moore, B., Rafalow, L., Ramberg, Y., Snir, Y., Westerinen, A., Chadha, R., Brunner, M., Cohen, R. and Strassner, J., "Policy Core Information Model Extensions", [draft-ietf-policy-pcim-ext-05.txt](#), October 2001 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-05.txt](#), October 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.

[IPSO] Kent, S., "U.S. Department of Defense Security Options for the Internet Protocol", [RFC 1108](#), November 1991.

[IPSEC] Kent, S., and Atkinson, R., "Security Architecture for the Internet Protocol", [RFC 2401](#), November 1998.

[DMTF] Distributed Management Task Force, <http://www.dmtf.org/>

[CIMCORE] DMTF Common Information Model - Core Model v2.6 which can be found at http://www.dmtf.org/standards/cim_schema_v26.php

[CIMUSER] DMTF Common Information Model - User-Security Model v2.6 which can be found at http://www.dmtf.org/standards/cim_schema_v26.php

[CIMNETWORK] DMTF Common Information Model - Network Model v2.6 which can be found at http://www.dmtf.org/standards/cim_schema_v26.php

14. 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.

15. Authors' Addresses

Jamie Jason
Intel Corporation
MS JF3-206
2111 NE 25th Ave.
Hillsboro, OR 97124
E-Mail: jamie.jason@intel.com

Lee Rafalow
IBM Corporation, BRQA/502

Jason, et al

Expires February-2003

[Page 68]

4205 So. Miami Blvd.
Research Triangle Park, NC 27709
E-mail: rafalow@watson.ibm.com

Eric Vyncke
Cisco Systems
Avenue Marcel Thiry, 77
B-1200 Brussels
Belgium
E-mail: evyncke@cisco.com

16. Full Copyright Statement

Copyright (C) The Internet Society (1999). All Rights Reserved.

This document and translations of it maybe copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THEINTERNET ENGINEERING TASK FORCE DISCLIAMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMAITON HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTEIS OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

