Internet Draft
draft-ietf-ipsec-spsl-01.txt
Expires January 2000

M. Condell, BBN
C. Lynn, BBN
J. Zao, BBN
July 1, 1999

Security Policy Specification Language

Status of this Memo

Abstract

This document describes the Security Policy Specification Language (SPSL), a language designed to express security policies, security domains, and the entities that manage the policies and domains. The syntax and semantics of the language are presented here. SPSL currently supports policies for packet filtering, IP Security (IPSec), and IKE exchanges, however, it may easily be extended to express other types of policies.

Table of Contents

1. Introduction

The Security Policy Specification Language (SPSL) is a vendor and platform independent language for specifying communication security policies, especially those controlling the use of IPSec and IKE protocols. As the use of firewalls with strong authentication and virtual private networks (VPNs) with level 2 and 3 encryption become more popular, the need for managing these security services and devices by means of security policies also becomes more acute. SPSL allows the security policies to be specified in an interoperable language, stored in common databases and processed by management systems distinguished from the security devices. As such, SPSL is a main component of a scalable policy based security management system [SPS].

The syntax of SPSL and several of its supporting object classes were derived from the Routing Policy Specification Language [RPSL]. However, the processing rules of SPSL are significantly different from those of RPSL. Although the language was designed initially for specifying IPSec and IKE policies, its flexible syntax allows it to be used to express stateless and stateful packet filtering rules. Moreover, the language is extensible: new object classes can be added for the purpose of specifying policies of other security or communication protocols.

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [Bra97].

1.1 Language Requirements

SPSL was designed to meet the following requirements:

* Support for IPSec/IKE and general communication security policy specification,

* Support for both node- and domain-based policy models,

* Support for multiple distributed policy enforcement points,

* Support for authentication and authorization mechanisms to aid policy management,

* Support for flexibility and extensibility of the language.

1.1.1 Specification of Security Policies

In SPSL, a policy is defined as a binding between a set of communication conditions and a corresponding set of security actions. This abstraction is used to specify communication security policies in general and IPSec/IKE policies in particular. If an on-going communication (or one to be established) matches one of the conditions then one of the prioritized alternative sets of actions must be taken

to protect the communication. This abstraction also captures current policy enforcement practices.

The set of communication conditions in a policy are specified as one or more tuples of selector values. This is because IPSec transports and tunnels depend on security associations that are attached to specific values of chosen communication parameters, known as the selectors. SPSL supports all the selectors mentioned in IPSec architecture document [Kent98] and a much extended collection as described in Section 5.4.

The actions of a policy can affect different communication security operations:

* They may specify simple packet filtering actions: discard the packet, pass it, or forward it to a designated network entity.

* They may specify security proposals necessary for protecting IKE exchanges.

* They may specify IPSec tunnels or transports for passing the packets. The possible security mechanisms to protect the tunnels and the transports are specified as IKE proposals as specified in the IPSec Domain of Interpretation [DOI].

SPSL supports IPSec policy data model [PolMod] proposed by Pereira and Bhattacharya in order to effect the last two types of actions.

1.1.2 Node- and Domain-Based Models

SPSL enables two ways to associate security policies with network entities, known as the node-based and the domain-based policy models.

In the node-based model, security policies are bound to individual network nodes and security devices, e.g., firewalls, hosts, etc. The policies associated with a network node specify the protection for the communications to and from the node. These policies are expected to be enforced by the node itself. The policies associated with a security device (formally known as a policy enforcement point) specify the protection for the communications passing through these agents. Either the source or the destination of the communication must be among the nodes that the agent is authorized/expected to protect. In this model, both the network nodes and the security policy enforcement agents manage their own policies.

In the domain-based model, security policies are bound to a security domain. A security domain is defined as a connected set of network entities that are protected by policy enforcement points (PEP) placed on every communication path going through the perimeter of the domain. Every policy enforcement point of the domain works to enforce the common set of security policies associated with the domain. Security domains may be completely disjoint, contained in one another,

comprised of several sub-networks, or just hosts that enforce their own policy. In this model, the policies associated with a domain are managed by one or more special agents common to the entire domain. These special agents act as policy servers. They may be distinct network entities or co-located with the nodes or the policy enforcement agents of the domain.

1.1.3 Multiple Distributed Policy Enforcement Points

SPSL allows explicit selection of enforcement points(s) of a security policy. The choices can be interfaces of end nodes, en-route security gateways (SG), e.g., firewalls, specified by IP addresses. The explicit selection of an enforcement agent allows a system to choose a communication path different than the one chosen by the routing infrastructure. This facility is especially useful for tunnel establishment and management.

1.1.4 Authentication and Authorization Mechanisms

SPSL has object classes to support the following security services:

1. data integrity, data origin authentication: every policy object is protected by using a public key signature. Both RSA [RSA] and DSA [DSA] signature algorithms are supported. This also offers non-repudiation proof of the issuer(s) of the policies.

2. authentication and authorization of policy management entities: management objects such as maintainers have public key certificates associated with them so that they may issue policies and/or identify themselves to a security management system for access control purposes.

With these services, users of SPSL policy specifications can always verify the integrity and the origin of the policies and allow only authorized personnel to maintain the policies.

1.1.5 Language Flexibility and Extensibility

SPSL is a flexible and extensible language. The language is flexible because its present syntax enables it to specify policies for different uses. For example, it can be used to specify non-cryptographic stateless packet filtering rules as well as IPSec tunnels for virtual private networks. It can also be used to effect standard IPSec or fine grain selector matching. In addition, it supports both node- and domain-based models.

The language is also extensible. It allows new object classes to be created by following a syntactic rule similar to inheritance. Consequently, the language can be extended for specifying policies of different communication and security protocols/applications.

1.2 Language Structure

SPSL uses the object paradigm although it is not an object-oriented language. The language defines a small set of classes, which can instantiate objects maintaining data relevant to policy specification. The data are contained in the attributes defined in the object classes. There are no executable methods in the classes nor do the classes form types.

New classes can be created as needed based on a syntactic rule similar to inheritance in object-oriented languages. For example, a new class may have all the attributes of an existing class in addition to its own attributes. However, the old and the new classes are not related by type polymorphic relations because the objects do not contain types. Objects in an SPSL file are distinguished and referred by the unique values of their first attribute, known as the key attribute.

1.2.1 Categories

SPSL is comprised of the following four categories:

- Primitive Data - contain basic or atomic data elements used in policy specification, e.g., object-name, ipv4-address, integer-range, date, etc.

- Management Agents - contain information relevant to the management entities; the existing classes in this category are maintainer (mntner) and certificate (cert).

- Network Entities - depict the network elements that are relevant to policy specification; the existing classes are node, node-set, gateway, gateway-set, polserver, and domain.

- Policies - contain the policy specification; there are only two classes at the moment: class policy specifies general packet filtering rules and class ipsec-policy specifies IPSec selectors and actions. Objects of the policy class may appear in two forms for short or long policy specification.

1.2.2 Class Design

Each class has a set of attributes which store information about the objects of the class. Attributes can be mandatory (man) or optional (opt). A mandatory attribute MUST be defined for all objects of the class, and an optional attribute MAY be omitted. Attributes can also be single valued (s-v) or multiple valued (m-v). A single valued attribute MUST only appear once per object. A multiple valued attribute MAY appear more than once per object. Each object is uniquely identified by the key attribute of its class.

An SPSL object is textually represented as a list of attribute-value pairs. An object's representation begins with the class-key attribute-value pair. Each attribute-value pair is written on a

separate line. The attribute name precedes the first colon, ":", and is followed by the value of the attribute. An attribute-value pair may span multiple lines. A "\" MUST be used as the last character of a line to indicate that the line is continued. An object's representation ends when a blank line (i.e., a line containing only whitespace characters such as spaces, tabs, and carriage returns) is encountered.

The order of attributes within a signed object is significant. The order of the written form of the attributes when signed must be preserved until the object is validated or resigned. This ordering is necessary to be able to verify signatures of objects. The class key must always be the first attribute. If the 'char-set' attribute is included, it must always precede any 'notes' attribute. The last attribute in any object must be the 'signature' attribute(s). If multiple 'policy' attributes are included in a single policy class object then their ordering must be preserved, unless the policy is being specifically changed. This is required since the ordering of policies may affect how they are applied (section 5.5).

A value of an attribute may be a single data item or a list of data items of the same type. A list is represented by separating the list members by commas ",". Note that the options of having a list of values and/or multiple values are two independent choices for an attribute. A multiple valued attribute may appear multiple times within an object, and the value in each occurrence may or may not be a list. A single valued attribute may also have a list value.

The default character set is ISO 8859-1 (Latin-1) [ISO8859]. The character set is an eight bit encoding where the lower 7 bits are identical to the ASCII character set. This default MUST always be used for all the attribute tags and attribute values. The character set for the notes attribute value MAY be overridden by using the 'char-set' attribute.

An object's specification may contain comments. A comment may appear anywhere in an object definition. It starts at the first "#" character on a line and ends at the first end-of-line character. The "\" character may be used to escape the comment character, so that it will be used as a "#" character and not a comment. "\\" will be used to represent the "\" character. Whitespace characters may be used to improve readability.

1.2.3 Naming Scheme and Scope

Since an SPSL object is distinguished by and referenced by its key attribute, the value of that attribute (which is usually a name) must be unique in the entire policy specification file (SPSL file). The actual scope of uniqueness may differ depending on the choice of policy model. In the node-based model, the names must be unique within a node or a security enforcement point that owns the policies.

In the domain-based model, the names must be unique within the set of

security servers that manage the policies of one or more domains in a primary-secondary server configuration. Note that the name of an object must be unique among all classes, not merely among its own class.

A recommended method for satisfying this uniqueness requirement is to adopt the following hierarchical naming scheme. A hierarchical object name is a sequence of names (usually, domain, node, or gateway names) separated by colons ":". The names are arranged following a descending order starting with the highest level name. For example, SG-BAZ:SG-BAR:SG-FOO is a valid hierarchical object name with SG-BAZ being the top level name.

1.2.4 $INCLUDE Extension

An SPSL file may actually consist of multiple files containing complete SPSL objects. One SPSL file may be included as part of another file using the following:

```
$INCLUDE <filename>
```

The contents of <filename> are included in the SPSL file at the exact place where the $INCLUDE line is in the SPSL file. As with SPSL objects, this line must be separated from other SPSL objects by a blank line.

2. Primitive Data Types

The following are the commonly used data types in SPSL. [Note: many of these data types are identical to those specified in RPSL. [RPSL]]

<object-name> All SPSL objects are identified by a name. An <object-name> is made up of letters, digits, the character underscore "_", the character period ".", the character colon ":", and the character hyphen "-"; the first character of a name must be a letter, and the last character of a name must be a letter or a digit. Names are case sensitive.

<filename> is made up of letters, digits, the character underscore "_", the character period ".", the character colon ":", the character hyphen "-", the character slash "/", and the character backslash "\". ("\\" must be used to represent a single "\". File names are case sensitive.

<ipv4-address> An IPv4 address represented as a sequence of four integers in the range from 0 to 255 separated by the character dot ".". For example, 172.17.128.5 represents a valid IPv4 address.

<ipv6-address> An IPv6 address represented as a sequence of eight hexadecimal integers in the range from 0 to FFFF separated by the character colon ":". The last two hexadecimal integers may be replaced with an <ipv4-address>. A single string of one or more hexadecimal integers with value zero (0) may be omitted.

For example, 129:0:0:0:5:800:20C2:F35B, 129:0:0:0:5:800:32.194.243.91, and 129::5:800:32.194.243.91 all represent valid IPv6 addresses, and all encode the same value.

<ip-address> An <ipv4-address> or <ipv6-address>.

<address-range> An address range is represented as an IP address followed by the character dash "-" followed by a second IP address, by an IP address followed by the keyword "mask" followed by a second IP address, or by an IP address followed by a slash "/" followed by an integer. The addresses MUST be either both <ipv4-address>'s or both <ipv6-address>'s. The dash form of an address range is inclusive. The following are valid address ranges: 172.16.1.1-172.16.1.200, 172.16.1.1-172.16.3.33. The mask form uses the second IP address to specify a bit mask. One bits in the mask correspond to bits in the address that may not vary. A valid masked address range is: 10.0.0.1 mask 255.255.0.255. The slash form uses the integer to indicate the number of bits in the address, beginning from the most-significant, that may not vary. A valid address range in this form is: 192.168.2.0/24.

<date> A date is represented as an eight digit integer of the form YYYYMMDD where YYYY represents the year, MM represents the month of the year (01 through 12), and DD represents the day of the month (01 through 31). For example, June 24, 1996 is represented as 19960624.

<integer-range> specifies an integer, minimum integer, maximum integer, or range of integer values. It uses the following syntax:

<integer> | min <integer> | max <integer> | <integer>-<integer>

The following are valid <integer-range>'s: 5, 67-100, min 50, max 60.

<phone-number> is a phone or fax number. A phone number may contain digits, spaces " ", plus "+", minus "-", and the letter "x" to indicate extension numbers. The following are valid <phone-number>s: +31 20 123-4676, +44 123 987654 x4711.

<email-address> is as described in RFC-822 [rfc822].

<dns-name> is as described in RFC-1034 [rfc1034].

<free-form> is a sequence of ASCII characters.

<X-name> is a name of an object of type X. That is <mntner-name> is a name of a mntner object.

<oid> is an object identifier of type <object-name>.

<or-address> is an X.400 address of type <free-form>. See Appendix in [PKIXP1] for further definition of the syntax.

<relative-distinguished-name> represents an X.500 distinguished name of type <free-form>. See Appendix in [PKIXP1] for further definition of the syntax.

<edi-party-name> EDI Party Name of type <free-form>. See Appendix in [PKIXP1] for further definition of the syntax.

<uri> Uniform Resource Identifier of type <free-form>.

<general-name> is of the form <name-type> <name>. <name-type> describes the type of name used in <name>. <name> is a string that identifies a name. Its format depends upon the <name-type>. The following name types and their corresponding <name> formats have been defined as follows (based on CRL Distribution Points extension in [PKIXP1]):

<name-type>	description of type	<name> format
other	Other Name	<oid> <freeform>
n822	RFC 822 Name	<email-address>
dns	DNS Name	<dns-name>
x400	X400 Address	<or-address>
dirname	Directory Name	list of <relative-distinguished-name>
ediname	EDI Party Name	<edi-party-name>
uri	Uniform Resource Identifier	<uri>
ipaddr	IP Address	<ip-address>
regid	Registered ID	<oid>

3. Management Agent Classes

The classes mntner and cert and the attributes mnt-by and changed in all classes contain information about the management agents of the policy specification. Among them, the mntner class specifies what entities can create, delete, and replace other objects. These classes do not specify security policies.

3.1 mntner Class

The mntner class defines entities that can create, delete, and replace SPSL objects. A provider, before creating SPSL objects, first needs to create a mntner object. The attributes of the mntner class are shown in Figure 1.

Attribute	Value	Type (Sect. 1.2.2)
mntner:	<object-name>	man, s-v, key
char-set:	<char-set>	opt, s-v

Attribute	Value	Type (Sect. 1.2.2)
notes:	<free-form>	opt, m-v
auth:	<scheme-id> <auth-info>	man, m-v
address:	<free-form>	man, m-v
phone:	<phone-number>	man, m-v
fax-no:	<phone-number>	opt, m-v
email:	<email-address>	man, m-v
mnt-by:	list of <mntner-name>	man, m-v
certs:	list of <cert-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description below	man, m-v

Figure 1: mntner Class Attributes

The 'mntner' attribute is mandatory and is the class key. Its value is an SPSL name. The 'auth' attribute specifies the scheme that will be used to identify and authenticate update requests from this maintainer. It has the following syntax:

```
auth: <scheme-id> <auth-info>

E.g.,
     auth: crypt-pw dhjsdfhruewf
```

The <scheme-id>'s currently defined are: "cert", "pgp", and "crypt-pw". The <auth-info> is additional information required by a particular scheme: in the case of "crypt-pw", it is a password in UNIX crypt format; and in the case of "pgp", it is a PGP public key; in the case of "cert", it is a list of <cert-name> to match the public key certificates in the 'cert' attribute. If multiple 'auth' attributes are specified, an update request satisfying any one of them is authenticated to be from the maintainer.

The 'char-set' attribute identifies the name of the character set used for the value of the notes attribute in this object. The 'char-set' does not apply to the attribute names; the default character set is always used for them. If this attribute is not included, then the default character set is used.

The 'address', 'phone', 'fax-no', and 'email' attributes provide contact information for the maintainer. The 'address' attribute SHOULD contain one complete address per instance of the attribute.

The 'notes' attribute contains a free-form textual description of the object and other notes about the object. The 'mnt-by' attribute is a list of mntner object names. The authorization for replacement or deletion of this object is governed by any of the maintainer objects referenced. The 'changed' attribute documents who last changed this object, and when the change was made. This attribute is multi-valued so that a history of who made changes and when MAY be kept. Only the most recent change MUST be kept. If multiple 'changed' attributes are

saved, then they MUST be ordered from most to least recent. The <mntner> identifies who made the change. <date> is the date of the change.

The 'certs' attribute lists certificate objects that point to the public key certificates for this mntner.

The 'signature' attribute contains a signature of the object. Signatures are computed over the textual representation of all the attributes in the object, except any 'signature' attributes. For purposes of the signature, all white-space is reduced to a single space, except for carriage returns which are included in the signature. Line continuation characters and the following carriage return are included in the signature. Comma separated lists do not have a space on either side of the comma "," for the signature. There SHOULD be at least one 'signature' line for each <mntner-name> in mnt-by. When an object is modified, all signature attributes MUST either be recomputed or removed from the object so that all signatures are valid. The attribute has the following syntax:

```
signature: <mntner-name> <cert-name>
           <signature-alg> <signature-data>

E.g.,
   signature: XYZ-IR-MNT XYZ-X509-CERT rsa-pkcs1 <signature-data>
```

The <mntner-name> and <cert-name> identify which mntner signed this object and which certificate was used. <signature-alg> is the algorithm used to create the signature. Currently the following signature algorithms are defined: "rsa-pkcs1", "dsa-sha1". <signature-data> is the signature that was generated.

Figure 2 shows an example mntner object. In the example, "cert" authentication is used.

```
mntner:      XYZ-IR-MNT
notes:       XYZ-IR Maintainer
auth:        cert XYZ-IR-X509-CERT
address:     XYZ Corp, 1 XYZ Place, Anytown, AS 12345, USA
phone:       +1 617 5551234
email:       jdoe@ir.xyz.com
mnt-by:      XYZ-IR-MNT
certs:       XYZ-IR-X509-CERT
changed:     XYZ-IR-MNT 19970820
signature:   XYZ-IR-MNT XYZ-IR-X509-CERT dsa-sha1 <signature-data>
```

Figure 2: An example mntner object.

The 'char-set', 'notes', 'mnt-by', 'changed', and 'signature' attributes are attributes of all SPSL classes. Their syntax, semantics, and type (mandatory, optional, multi-valued, or single-valued) are the same for for all SPSL classes. They are not discussed further in the remaining sections.

3.2 cert Class

The cert class identifies a public key certificate that may be used to sign SPSL objects. A cert object either specifies a certificate or the location of a certificate. The 'cert' attribute identifies the name of the object.

Attribute	Value	Type (Sect. 1.2.2)
cert:	<object-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
certificate:	see description below	opt, s-v
certlocation:	see description below	opt, m-v
crllocation:	see description below	opt, s-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description in Section 3.1	man, m-v

Figure 3: cert class attributes

The 'certificate' attribute has the following syntax:

```
certificate:  <cert-type> <cert-data>
```

<cert-type> describes the type of certificate represented. Currently the following types are defined: "pkcs7", "pgp", "dnskey", "x509_sig", "x509_ke", "kerberos", "spki". <cert-data> is the actual certificate described by this object, encoded as a hexadecimal representation of the certificate.

certificate type	description
pkcs7	PKCS #7 wrapped X.509 certificate
pgp	PGP certificate
dnskey	DNS signed key
x509_sig	X.509 certificate - signature
x509_ke	X.509 certificate - key exchange
kerberos	Kerberos tokens
spki	SPKI certificate

The 'certlocation' attribute has the following syntax:

```
certlocation: <cert-type> <fetch-protocol>
              <general-name> | filename <filename> |
              rdn <relative-distinguished-name>
```

<cert-type> is as defined above. <fetch-protocol> specifies the preferred protocol should be used to fetch the certificate from this location. Currently the following protocols have been defined: "cdp", "dns", "file". The location of the certificate is identified either by using a <general-name> or a <relative-distinguished-name> when fetching with CDP or DNS, or a <filename> when fetching from a locally stored file.

The 'crllocation' attribute indicates where a certificate revocation list (CRL) may be found for this certificate. It has the following syntax:

```
crllocation: <crl-type> <fetch-protocol>
          <general-name> | filename <filename> |
          rdn <relative-distinguished-name>
```

This is similar to the certlocation attribute, except that <crl-type> is used in place of <cert-type>. <crl-type> describes the type of CRL that may be found at this location. Currently, the following CRL type has been defined: "x509".

At least one 'certificate' or 'certlocation' attribute MUST be present in a cert object. It is possible for a 'certificate' and a 'certlocation' attribute, or multiple 'certlocation' attributes to be present in a single cert object, but they SHOULD all refer to the same certificate, otherwise the wrong certificate may be used.

4. Network Entity Classes

4.1 node Class

The node class identifies a set of interfaces on a network entity that may have policies associated with them. This definition allows a single network entity to be represented by one or more node objects. It also allows policies to be associated with specific interfaces or addresses of a network entity.

Attribute	Value	Type (Sect. 1.2.2)
node:	<node-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
name:	<dns-name>	man, s-v
alias:	<dns-name>	opt, m-v
ifaddr:	<ip-address>	man, m-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description in Section 3.1	man, m-v

Figure 4: node class attributes

The 'node' attribute is the class key, which uniquely identifies the node object.

The 'name' attribute is a valid DNS name identifying the network entity to which the interfaces in the object are attached. Each 'alias' attribute, if present, should be a canonical DNS name of the network entity. The 'ifaddr' attribute specifies the IP address of each interface of the node.

Figure 5 shows two examples of node objects.

```
node:    SQUATCH
name:    squatch.foo.com
ifaddr:  172.16.3.11
ifaddr:  192.2.1.83

node:    SG-FOO-FIREWALL:COTTON
name:    cotton.foo.com
ifaddr:  172.16.5.196
```

Figure 5: node object examples

4.2 node-set Class

The node-set class provides a means to group several nodes into one object. The class may be used to group together the interfaces of a single host or of multiple hosts. The nodes in a node-set object are expected to contain the interfaces of a common set of network entities.

The node-set class is defined below:

Attribute	Value	Type (Sect. 1.2.2)
node-set:	<node-set-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
members:	list of <node-names> and/or <node-set-names>	man, m-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description in Section 3.1	man, m-v

Figure 6: node-set class attributes

The 'node-set' attribute is the class key, which uniquely identifies the node-set object. The 'members' attribute is a list of the node objects and node-set objects which are grouped by the node-set object.

4.3 gateway Class

The gateway class identifies a set of interfaces on a policy enforcement agent, e.g., a security gateway, that can enforce the security policies associated with the enforcement agent or the domain for which it enforces policy.

Attribute	Value	Type (Sect. 1.2.2)
gateway:	<gateway-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
name:	<dns-name>	man, s-v
alias:	<dns-name>	opt, m-v
ifaddr:	<ip-address>	man, m-v
preference:	<integer>	man, s-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description in Section 3.1	man, m-v

Figure 7: gateway class attributes

The 'gateway' attribute is the class key, which uniquely identifies the gateway object.

The 'name' attribute is a valid canonical DNS name identifying the network entity on which the policy enforcement agent is implemented. Each 'alias' attribute, if present, should be a canonical DNS name of the network entity. The 'ifaddr' attribute specifies the IP address of an interface.

The 'preference' attribute gives a hint as to the preference of routing to use this gateway. 1 is the highest preference and the preference decreases as the integer increases. This is only used for purposes of the domain object and is explained further in section 4.6.

Figure 8 shows two examples of gateway objects.

```
gateway:    SG-FOO-FIREWALL
name:       foo-firewall.foo.com
ifaddr:     172.16.0.1
ifaddr:     192.2.1.83
preference: 1

gateway:    SG-FOO-FIREWALL:SG-IS-FIREWALL
name:       is-firewall.foo.com
ifaddr:     172.16.5.196
preference: 3
```

Figure 8: gateway object examples

4.4 gateway-set Class

The gateway-set class provides a means to group gateways. It can be used to group together the interfaces of a single gateway or the interfaces of multiple gateways spread across several gateway objects, so that they may be referred to as a single object.

The gateway-set class is defined below:

Attribute	Value	Type (Sect. 1.2.2)
gateway-set:	<gateway-set-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
members:	list of <gateway-names> and/or <gateway-set-names>	man, s-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description in Section 3.1	man, m-v

Figure 9: gateway-set class attributes

The 'gateway-set' attribute is the class key, which uniquely identifies the gateway-set object. The 'members' attribute is a list of gateway objects and/or gateway-set objects which are grouped by the gateway-set object.

4.5 polserv Class

The polserv class defines the policy servers that are capable of managing security policies.

Attribute	Value	Type (Sect. 1.2.2)
polserv:	<policy-server-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
name:	<dns-name>	man, s-v
alias:	<dns-name>	opt, m-v
ifaddr:	<ip-address>	man, m-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description in Section 3.1	man, m-v

Figure 10: polserv class attributes

The 'polserv' attribute is the class key, which uniquely identifies the policy server object. The 'name' attribute is a valid DNS name identifying the network entity on which the policy server is located. Each 'alias' attribute, if present, should be a canonical DNS name of the network entity. The 'ifaddr' attribute specifies the IP address of an interface of the policy server.

Figure 11 shows a simple example of a policy server object.

```
polserv: PS-SECURITY
name:    foo-pol-server.foo.com
ifaddr:  172.16.0.2
```

Figure 11: polserv object example

4.6 domain Class

The domain class provides a means to define a security domain, which is a cluster of network entities protected by a common set of security policies which are enforced by the policy enforcement pointss located at the perimeter of the domain.

A security domain is the basic topological structure for a domain-based security model [Section 1.1.2]. It consists of three components:

1. Coverage - a security domain must be authorized to include a specific set of network entities. That specification is provided in the 'coverage' attribute, and can take the form of a list of IP addresses, a list of IP address ranges, a list of nodes, and/or a list of node-sets.

2. Policy Enforcement Points - the network entities included in a security domain are protected by a set of policy enforcement points located at the perimeter of the domain. The policy enforcement points are specified by the 'gateways' attribute, which may contain a list of gateways or gateway-sets. The gateways in the list MAY be ordered using the 'gateways' 'preference' attribute.

3. Policy Servers - one or more policy servers are assigned to the security domain to manage the security policies of the domain. These servers are given in a list under the 'polservs' attribute. The first member of the list MUST be the primary server, and the rest are any secondary servers.

With these three components, the domain class is defined below. Individual domain objects are uniquely identified by the 'domain' attribute, which is the class key.

Attribute	Value	Type (Sect. 1.2.2)
domain:	<domain-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
coverage:	[list of <ip-address > [,]] [list of <address-range> [,]] [list of <node-name> [,]] [list of <node-set-name>]	man, m-v
gateways:	list of <gateway-name> and/or <gateway-set-name>	man, s-v
polservs:	list of <policy-server-names>	man, s-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description in Section 3.1	man, m-v

Figure 12: domain class attributes

5. Policy Class

A policy class object specifies a binding between a set of communication conditions and a set of actions.

In the current version of SPSL, two policy classes are defined. The general class described in sections 5.1 and 5.2 defines the conditions and a transfer action that allows for specifying packet filtering rules. The class described in Section 5.3 is to be used for specifying IPSec and IKE policies.

Moreover, objects of the general policy class may take one of two possible formats. The short format expresses the policy in a single 'policy' attribute and the long format expresses each part of a policy in a distinct attribute. Each of the two formats is appropriate for different applications. They will be discussed in the next two sections, along with comments on their strengths and weaknesses.

Both formats of the policy class share three attributes. Figure 13 shows the class definition (in short format) in order to display the common attributes. Among them, 'policy-name' gives the name of the policy. The 'cache-expiry' attribute indicates, in seconds, the maximum time that this policy should be cached. It can be regarded as a hint to any entities that may cache this policy. If the attribute is absent or has a value of zero then no expiration time is suggested.

The 'association' attribute specifies the names of one or more nodes, gateways, or domains that own the policy. If a node-based or domain-based policy model is being used, strict rules of association must be observed depending on the model. In the node-based model, a policy can be associated with an object from the node, node-set, gateway, or gateway-set classes but never with a domain object. In the domain-based model, a policy can be associated with an object from the node, node-set, or domain classes but not with a gateway or gateway-set object. This is because the policy associated with a gateway or gateway-set object will be enforced by that particular object instead of by all of the enforcement agents of a specific domain. However, a node or node-set object is allowed its own policies in a domain-based association because the object may be regarded as a single/multiple node domain.

5.1 policy Attribute (Short Format)

The short format of the policy class specifies the policy in a single 'policy' attribute that is structured as follows:

Attribute	Value	Type (Sect. 1.2.2)
policy-name:	<object-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
association:	<node-name> | <node-set-name> | <gateway-name> | <gateway-set-name> | <domain-name>	man, s-v
cache-expiry:	<integer>	opt, s-v
policy:	as described below	opt, m-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description in Section 3.1	man, m-v

Figure 13: policy class attributes, short format

```
policy: dst * | any | list of [not] <ip-address>
              | list of [not] <address-range>
             [port * | opaque | any | list of [not] <port>
                     | list of [not] <port-range>
                  [dynamic [<port-range>]]]
        [src * | any | list of [not] <ip-address>
               | list of [not] <address-range>
             [port * | opaque | any | list of [not] <port>
                     | list of [not] <port-range>
                  [dynamic [<port-range>]]]]
        [xport-proto * | opaque | any | list of [not] <proto>
                       | list of [not] <proto-range>]
        [direction inbound | outbound [, symmetric]]
        permit [, forward <dest>] | deny [, forward <dest>]
             | forward <dest>
```

The "dst" tag specifies a list of <ip-address>s or <address-range>s to which this policy does (or does not) apply. The address may be specified as "any" or the wildcard, "*", to indicate this applies to traffic destined to all addresses. Otherwise, the address is a list of individual IP addresses, or address ranges specified by a minimum and maximum address (inclusive), or address ranges specified using an address and mask.

An address may be preceded by the qualifier "not" to indicate that the address from a packet must not be the one specified. Note that it not useful to include some list members with the "not" qualifier and some without it. For a distinct X and Y, an expression "X or not Y" is equivalent to just "not Y". An expression "X and not Y" is equivalent to just "X". Note the special case where a list contained the same address both with and without "not" -- those two list members are equivalent to "any". Consequently, when a list contains no "not" qualifiers, the interpretation is "X or Y or Z", while if each list member has a "not" qualifier, the list is interpreted as "not X and not Y and not Z".

Attribute "dst" may optionally be followed by "port" and a list of destination port numbers or ranges of port numbers to which this policy does (or does not) apply. Additionally, "port" may be followed by the tag "dynamic" and an optional range of port numbers. This specifies that a connection established by using one of the port numbers following the "port" tag, may then use dynamic ports for the rest of the communications using that connection. If a range of port numbers follows the dynamic tag, then dynamic ports are only allowed within that specified range. If the range is not specified, the port range defaults to "*". If the dynamic tag is not used, then dynamic ports are excluded from this policy.

A source address and port(s) may optionally be specified in a similar manner using the "src" tag. The source address and source and destination ports default to "*" if they are not specified.

The transport protocol may be specified using the optional tag "xport-proto", which defaults to "*" if not specified. The transport protocol may be specified as a single transport protocol number, a list of protocol numbers, or a range of protocol numbers in the form <number>-<number>. It may also be specified as "*", "any", or "opaque". Note that when a port is specified as described in the previous two paragraphs, then the protocol associated with those ports must be specified using the "xport-proto" phrase.

The "direction" specification is used to specify whether a packet is entering the domain associated with the policy (inbound) or exiting it (outbound). If the optional qualifier "symmetric" is present, a second policy will automatically be created with the direction sensitive fields -- "src" and "dst", and src port and dst port -- switched.

The transfer action of "permit" or "deny" must be specified to indicate whether packets that match this policy should be passed or dropped, respectively. The transfer action may additionally specify that the matching packets be forwarded to a specified destination, e.g., a policy server. The destination may be specified by either a DNS name, preceeded by "dns", or by an IP address.

```
policy-name: foo
association: sg-bar
policy:      dst 172.16.0.0-172.16.255.255
             src 192.168.100.0-192.168.100.255
             xport-proto 6 permit
policy:      dst 172.16.0.0-172.16.255.255 deny
```

Figure 14: policy object example, short format

In this example, this policy denies all packets destined to IP addresses from 172.16.0.0 to 172.16.255.255, unless they are from addresses 192.168.100.0 to 192.168.100.255 and use TCP. Note that the ordering of the 'policy' attributes is important (see section 5.5).

5.2 policy Attribute (Long Format)

The long format policy class makes each part of the policy attribute an explicit attribute.

Attribute	Value	Type (Sect. 1.2.2)
policy-name:	<object-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
association:	<node-name> | <node-set-name>	
	| <gateway-name> | <gateway-set-name>	
	| <domain-name>	man, s-v
cache-expiry:	<integer>	opt, s-v
valid-period:	list of <valid-time>	opt, m-v
dst:	see below	opt, m-v
src:	see below	opt, m-v
xport-proto:	see below	opt, m-v
direction:	inbound | outbound [, symmetric]	opt, s-v
userid:	* | any | list of [not] n822 <email-addr>	
	| list of [not] dn <distinguished-name>	opt, m-v
systemname:	* | any | list of [not] <general-name>	
	| list of [not] dn <distinguished-name>	opt, m-v
ipv6-class:	* | any | list of [not] <integer-range>	opt, m-v
ipv6-flow:	* | any | list of [not] <integer-range>	opt, m-v
ipv4-tos:	* | any | list of [not] <integer-range>	opt, m-v
seclabel:	* | any | list of [not] <seclabel>	opt, m-v
see Section 5.4 for additional selectors		opt, m-v
tfr-action:	see below	opt, m-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:	see description in Section 3.1	man, m-v

Figure 15: policy class attributes, long format

The attributes are specified as follows:

```
dst: * | any | list of [not] <ip-address>
       | list of [not] <address-range>
     [port * | opaque | any | list of [not] <port>
             | list of [not] <port-range> [dynamic [<port-range>]]]

src: * | any | list of [not] <ip-address>
       | list of [not] <address-range>
     [port * | opaque | any | list of [not] <port>
             | list of [not] <port-range> [dynamic [<port-range>]]]

xport-proto: * | opaque | any | list of [not] <proto>
               | list of [not] <proto-range>

tfr-action: permit [, forward <dest>] | deny [, forward <dest>]
            | forward <dest>
```

```
   <valid-time>: [year yyyy-yyyy] [month 000000000000]
        [day-of-month 00000000000000000000000000000000]
        [day-of-week 0000000] [time [not] hh:mm:ss-hh:mm:ss]
```

The 'valid-period' attribute describes one or more time periods in which the policy is valid. If more than one time period is expressed in an object, then the periods are related by a logical OR. Each time period consists optionally of a range of years, bitmask of months, bitmask of days-of-the-month and and -of-the-week, and a time period. These fields of a time period are related by a logical AND.

The "year" is a range of years to which the policy applies. The "month" is a 12-bit bitmask of the months with January as the first bit and December the last bit. If a bit is set to 1, the policy is valid during that month, if 0, it is not valid during that month. The "day-of-month" is a 31-bit bitmask of the days-of-the-month with 1 as the first bit and 31 the last bit. If a bit is set to 1, the policy is valid during that day, if 0, it is not valid during that day. The "day-of-week" is a 7-bit bitmask of the days-of-the-week with Sunday as the first bit and Saturday the last bit. If a bit is set to 1, the policy is valid during that day-of-the-week, if 0, it is not valid during that day. The "time" describes a range of times during which the policy is (or is not) valid. The times use a 24-hour clock and their values MUST be expressed in Greenwich Mean Time (Zulu). If any period is not described that field is interpreted as "any."

Attributes 'dst', 'src', 'xport-proto', 'direction', and 'tfr-action' (transfer action) are similar to their counterparts in the short format of the policy class. Note that the interpretation of a single selector attribute with a list value is similar to having each list member be a separate instance of the selector attribute since multiple occurances of these selectors (unlike the 'policy' selector) are interpreted as logical ORs.

The direction MUST be specified either in the 'policy' attribute or the 'direction' attribute. The 'tfr-action' attribute specifies an action that MUST be taken, as specified above.

The 'user-id' attribute specifies either an email address or a distinguished name to identify a particular user. The 'systemname' attribute uses a DNS name, an X.500 general name, or an X.500 distinguished name to identify a particular system.

The 'seclabel' attribute is used to identify an implementation specific security label. This should correspond to the implementation of the security level selector in IPSec. This attribute is a good example of the difference between "*" and "any". When "any" is used, the packet must contain the field which contains the selector value. When "*" is used, the packet does not have to have that field. Thus "any" means that the packet must specify a security label, but its value is not of interest. A "*" would mean that a packet need not contain any security label. The value "opaque" is used to match packets for which a selector field cannot be found, typically due to compression, fragmentation, or confidentiality.

Attributes 'ipv6-flow' and 'ipv6-class' specify a list of integers or integer ranges, optionally preceeded by "not", corresponding to the IPv6 flow label and transport class fields in the IPv6 header. 'ipv4-tos' is a list of integers or integer ranges, optionally preceeded by "not", corresponding to the IPv4 type of service field in an IPv4 header. These attributes default to "*" if they are not included. The 'tfr-action' and 'dst' attributes are mandatory, if the policy class is used in this format.

In order to represent the policies described in the above example (Figure 14), two policy objects must be created. Note that the ordering of the policy objects is important (see section 5.5).

```
policy-name:    baz
association:    sg-bar
dst:            172.16.0.0-172.16.255.255
src:            192.168.100.0-192.168.100.255
xport-proto:    6
tfr-action:     accept

policy-name:    foo
association:    sg-bar
dst:            172.16.0.0-172.16.255.255
tfr-action:     deny
```

Figure 16: policy object example, long format

Generally, policy objects will use one of the two formats, but it is possible to combine the features of both. The combined policy class looks as follows:

Attribute	Value	Type (Sect. 1.2.2)
policy-name:	<object-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
association:	<node-name> | <node-set-name>	
	| <gateway-name> | <gateway-set-name>	
	| <domain-name>	man, s-v

Figure 17: policy class attributes, combined format

```
   Attribute      Value                                  Type (Sect. 1.2.2)

   cache-expiry:  <integer>                                   opt, s-v
   valid-period:  list of <valid-time>                        opt, m-v
   policy:        as described above                          opt, m-v
   dst:           as described above                          opt, m-v
   src:           as described above                          opt, m-v
   xport-proto:   as described above                          opt, m-v
   direction:     inbound | outbound [',' symmetric]          opt, s-v
   userid:        * | any | list of [not] n822 <email-addr>
                | list of [not] dn <distinguished-name>       opt, m-v
   systemname:    * | any | list of [not] <general-name>
                | list of [not] dn <distinguished-name>       opt, m-v
   ipv6-class:    * | any | list of [not] <integer-range>     opt, m-v
   ipv6-flow:     * | any | list of [not] <integer-range>     opt, m-v
   ipv4-tos:      * | any | list of [not] <integer-range>     opt, m-v
   seclabel:      * | any | list of [not] <seclabel>          opt, m-v
      see Section 5.4 for additional selectors                opt, m-v
   tfr-action:    as described above                          opt, m-v
   mnt-by:        list of <mntner-name>                       man, m-v
   changed:       <mntner-name> <date>                        man, m-v
   signature:     see description in Section 3.1              man, m-v
```

Figure 17 (Cont'd): policy class attributes, combined format

If the 'policy' attribute is specified and any of the other attributes are also specified, those others apply to all the policy lines in this object. This holds true for sub-classes of the policy class, too. If a policy object has a conflict between a part of the policy attribute and one of the other attributes specified, it is an invalid object.

Figure 18 illustrates the combination of the two formats. The first policy object uses the combined format of the policy class. It has two 'policy' attributes and an 'xport-proto' attribute. The 'xport-proto' attribute is applied as part of the policy described by each of the 'policy' lines. This is equivalent to explicitly listing the 'xport-proto' attribute in each policy line, as shown in the second policy object.

```
policy-name:    tcp-foo
association:    sg-bar
policy:         dst 172.16.0.0-172.16.255.255
                src 192.168.100.0-192.168.100.255 accept
policy:         dst 172.16.0.0-172.16.255.255 deny
xport-proto:    6
```

This is equivalent to:

```
policy-name:    tcp-foo
association:    sg-bar
policy:         dst 172.16.0.0-172.16.255.255
                src 192.168.100.0-192.168.100.255
                xport-proto 6 accept
policy:         dst 172.16.0.0-172.16.255.255 xport-proto 6 deny
```

Figure 18: policy object example, combined format

While both formats allow the same policies to be specified, they each have their advantages and disadvantages. The short format allows uncomplicated policies, such as general default policies, to be specified in a compact format since it allows multiple policies to be defined in a single object. The short format, however, is not capable of specifying complex policies. The long format allows complex policies to be specified in a more straightforward manner. Also, the ability to combine both formats of the policy class, allows greater flexibility in how policies may be defined. Correct specification of policies is made easier by being able to specify those policies in a straightforward manner.

5.3 ipsec-policy Class

The ipsec-policy class is a sub-class of the policy class. It is used to state IPSec policies specifying whether or not AH or ESP are required for a particular communication, and the choice of security mechanisms to be used with IPSec protocols. It also specifies the security mechanisms that may be negotiated by IKE using the IPSec DOI [DOI]. Since it is a sub-class of the general policy class, it inherits attributes from the policy class. The inherited attributes are marked with an "*" in Figure 19 below.

Attribute	Value	Type (Sect. 1.2.2)
*policy-name:	<object-name>	man, s-v, key
*char-set:	<char-set>	opt, s-v
*notes:	<free-form>	opt, m-v
*association:	<node-name> | <node-set-name>	
	| <gateway-name> | <gateway-set-name>	
	| <domain-name>	man, s-v
*cache-expiry:	<integer>	opt, s-v
*valid-period:	list of <valid-time>	opt, m-v
*policy:	as described above	opt, m-v
*dst:	as described above	opt, m-v
*src:	as described above	opt, m-v
*xport-proto:	as described above	opt, m-v
*direction:	inbound | outbound [',' symmetric]	opt, s-v

Figure 19: ipsec-policy class attributes

```
   Attribute      Value                                Type (Sect. 1.2.2)

   *userid:       * | any | list of [not] n822 <email-addr>
                | list of [not] dn <distinguished-name>   opt, m-v
   *systemname:   * | any | list of [not] <general-name>
                | list of [not] dn <distinguished-name>   opt, m-v
   *ipv6-class:   * | any | list of [not] <integer-range> opt, m-v
   *ipv6-flow:    * | any | list of [not] <integer-range> opt, m-v
   *ipv4-tos:     * | any | list of [not] <integer-range> opt, m-v
   *seclabel:     * | any | list of [not] <seclabel>      opt, m-v
   *  see Section 5.4 for additional selectors           opt, m-v
   *tfr-action:   as described above                     opt, m-v
   ipsec-action:  see below                             opt, m-v
   ike-action:    see below                             opt, m-v
   *mnt-by:       list of <mntner-name>                 man, m-v
   *changed:      <mntner-name> <date>                  man, m-v
   *signature:    see description in Section 3.1        man, m-v
```

Figure 19 (Cont'd): ipsec-policy class attributes

```
   ike-action: ikemode <ikemode> pfs <usepfs>
               auth <auth-method>
               cipher <ikecipher> hash <hashalg>
               [group-desc <group-desc> |
                group-type <group-type> <hex-string> <hex-string>
                    <hex-string> <hex-string> <hex-string> <hex-string>]
               [prf <integer>] [field <integer>]
               expiry ( seconds | kilobytes ) <integer-range>

   where <ikemode> is one of: "aggressive", "main", "quick"

         <usepfs> is either "false" or "true"

         <auth-method> is one of "any",  "pre-shared", "dss",
                  "rsa", "rsa-encrypt", "rsa-revised" or an
                  <integer> as defined in [rfc2409], optionally
                  preceded by "not"

         <ikecipher> is a list of one or more of:
                  <ikecipheralg> [keylen <integer-range>]

         <ikecipheralg> is one of "any", "blowfish", "cast",
                  "des", "des3", "idea", "rc5", or an
                  <integer> as defined in [rfc2409], optionally
                  preceded by "not"

         <hashalg> is "any", or one or more of: "md5", "sha1", "tiger",
                  or an <integer> as defined in [rfc2409], optionally
                  preceded by "not"

         <group-desc> is one of: "modp-768", "modp-1024", "ec2n-155",
                  "ec2n-185", or an <integer> as defined in [rfc2409].
```

```
        <group-type> is one of: "modp", "ecp", "ec2n", or an
                 <integer> as defined in [rfc2409].

   ipsec-action:  [ esp <proposal-choice> cipher <ipseccipher>
                    [integrity <ipsecintegrity>]
                    [expiry ( seconds | kilobytes ) <integer-range>]
                    [tunnel | transport]
                    [from <location> [, <location>]]
                    [to <location> [, <location>]]
                  ]
                  [ ah <proposal-choice> integrity <ipsecintegrity>
                    [expiry ( seconds | kilobytes ) <integer-range>]
                    [tunnel | transport]
                    [from <location> [, <location>]]
                    [to <location> [, <location>]]
                  ]
                  [ ipcomp <proposal-choice> <ipcompalg> ]

   where <proposal-choice> is one of "req", "opt"

        <ipseccipher> is a list of one or more of:
                 <ipseccipheralg> [keylen <integer-range>]
                 [rounds <integer-range>]

        <ipseccipheralg> is one of "any", "blowfish", "cast",
                 "des", "des3", "idea", "idea3", "none", "rc4",
                 "rc5", "rfc1829-iv32", "rfc1829-iv64", or an
                 <integer> as defined in [rfc2407], optionally
                 preceded by "not"

        <ipsecintegrity> is a list of one or more of:
                 <integrityalg> [keylen <integer-range>]

        <integrityalg> is one of "any", "hmacdes","hmacmd5",
                 "hmacsha1", "kpdk", or an <integer> as defined
                 in [rfc2407], optionally preceded by "not"

        <ipcompalg> is "any", or one or more of: "deflate", "lzs",
                 "oui", or an <integer> as defined in [rfc2407],
                 optionally preceded by "not"

        <location> is "any", or one or more of: "dest", "host",
                 "local-sg", "remote-sg", <ip-address>,
                 "dns" <dns-name>
```

Two action attributes, 'ipsec-action' and 'ike-action', in addition to the policy class attributes, form the ipsec-policy class.

The 'ipsec-action' attribute specifies ESP, AH, and IP compression proposals that must be used to protect this communication. Each proposal may be either a required, "req," or optional, "opt," part of the specified communication. If a proposal is not included in the 'ipsec-action' it is prohibited.

If an ESP proposal is specified, the cipher algorithm to use is specified by the "cipher" tag and "any", a string describing a cipher algorithm, or a number corresponding to a cipher algorithm as defined in [rfc2407]. Each cipher algorithm may be further defined by an optional key length and number of rounds, if the cipher requires it. An integrity algorithm and its key length may optionally be specified. It defaults to "any" if not specified. Values are defined in [rfc2409]. The SA life type and life time may be specified with the "expiry" tag. If not used, the values default as described in section 4.5 of [rfc2407]. Tunnel or transport mode may be specified with the "tunnel" or "transport" tags. If neither are specified, either may be used. The end points of the SA may be specified with the "to" and "from" tags which are described in detail below.

If an AH proposal is specified, the integrity algorithm to use is specified by the "integrity" tag and "any", a string describing an integrity algorithm, or a number corresponding to an integrity algorithm as defined in [rfc2407]. Values are defined in [rfc2409]. The key length for each algorithm may also be specified. The SA life type and life time may be specified with the "expiry" tag. If not used, the values default as described in section 4.5 of [rfc2407]. Tunnel or transport mode may be specified with the "tunnel" or "transport" tags. If neither are specified, either may be used. The end points of the SA may be specified with the "to" and "from" tags which are described in detail below.

If an IP compression proposal is specified, the compression algorithm to use is specified by the "ipcomp" tag and "any", a string describing a compression algorithm, or a number corresponding to a compression algorithm as defined in [rfc2407]. The SA life type and life time may be specified with the "expiry" tag. If not used, the values default as described in section 4.5 of [rfc2407].

The "to" and "from" tags identify the endpoints (policy enforcement points) of the security association that the proposal describes. A network node may be explicitly specified as an endpoint using either its IP address or its DNS name. This node MUST be used as the specified endpoint of the SA. The endpoints of the SA may also be specified using a generic specification that allows the policy decision points to determine at which enforcement point to end the SA. "any" allows any enforcement point to be chosen as long as it is not

the same as the other endpoint. "host" specifies the appropriate (i.e. source or destination) endpoint of the communication. "dest", "local-sg" and "remote-sg" still need to be further thought and definition. [***] If the SA endpoints are not specified, they default to the source and destination endpoints specified in the policy object.

The 'ike-action' attribute specifies attributes that may be negotiated during IKE pahase one and whether perfect forward secrecy must be used in quick mode. The "ikemode" tag specifies whether IKE should use this specification in main, aggressive, or quick mode. The "pfs" tag specifies if perfect forward secrecy should be used. "auth", "cipher", and "hash" describe the authentication method, encryption algorithm, and hash algorithm to be used. These may be specified by "any", a string describing the appropriate algorithm, or a number corresponding to an algorithm as defined in [rfc2409]. A key length for the encryption algorithm may optionally be specified with each cipher algorithm using the "keylen" tag. A predefined group or a user-defined group may optionally be specified. A predefined group uses the "group-desc" tag followed by a string describing the group, or a number corresponding to a group as defined in [rfc2409]. A user-defined group uses the "group-type" tag followd by the string or number describing a group type (as defined in [rfc2409]) and the group description: the group prime/irreducible polynomial, group generator 1, group generator 2, group curve A, group curve B, and group order. A psuedo-random function may be specified with "prf" and the field size of a Diffie Hellman group may be specified with "field" and the size in bits. Finally, the life time and life type must be specified using the "expiry" tag.

If multiple 'ipsec-action' attributes or multiple 'ike-action' attributes are specified, they should be taken as logical ORs.

5.4 Selectors and Actions

SPSL policies all contain two types of attributes: selectors and actions. Selectors are the policy attributes that are used to match packets with a particular policy. Currently, all the selectors that are defined are contained in the base policy class, though sub-classes may also contain additional selectors. The selectors currently defined in the IPSec DOI are:

src	dst
src-port	dst-port
xport-proto	userid
systemname	ipv6-class
ipv6-flow	ipv4-tos
seclabel	direction

An extended list of selectors supported by SPSL includes:

ah-hdr	ipcomp-hdr	rhv1-dst
ah-nhdr	ipcomp-nhdr	rhv1-nhop
direction	iphdr	rhv1-phop
dop-hdr	ipv4-dst	seclabel
dop-nhdr	ipv4-frgm	src
dst	ipv4-frgo	src-port
dst-port	ipv4-hdr	systemname
esp-hdr	ipv4-hlen	tcp-ack
esp-nhdr	ipv4-id	tcp-dato
frag-hdr	ipv4-opt-lsrr-dst	tcp-dst-port
frag-nhdr	ipv4-opt-ssrr-dst	tcp-fin
hop-hdr	ipv4-prot	tcp-hdr
hop-nhdr	ipv4-src	tcp-psh
icmp4-code	ipv4-tlen	tcp-rst
icmp4-gwy	ipv4-tos	tcp-src-port
icmp4-hdr	ipv4-ttl	tcp-syn
icmp4-id	ipv6-class	tcp-urg
icmp4-mtu	ipv6-dst	tcp-urgp
icmp4-seq	ipv6-flow	udp-cks
icmp4-type	ipv6-hdr	udp-dst-port
icmp6-code	ipv6-nhdr	udp-hdr
icmp6-hdr	ipv6-src	udp-id
icmp6-id	ipver	udp-src-port
icmp6-mtu	rh-hdr	userid
icmp6-seq	rh-nhdr	xport-prot
icmp6-type	rh-vers	

Actions are the policy attributes that are applied to outbound packets and are used to decide whether or not to accept inbound packets. The actions currently defined in SPSL are:

tfr-action	ipsec-action	ike-action

5.5 Policy Order

Multiple policy objects and attributes are likely to apply to a particular communication. For example, most systems will have a default policy to deny all inbound communications. There will then be some more policies to permit specific inbound communications. A set of selector values (see section 5.4) that match one of the specific policies will also match the general default policy. SPSL must establish a rule so that the correct policy is applied to the communication. The rule must always provide the same answer when applied to the same set of policies, otherwise inconsistent policy enforcement may occur.

SPSL uses a simple rule to determine which policy should be applied to the on-going communication - physical ordering of the policies. The policy applied should always be the first policy that matches all the selectors of the communication. This ordering holds for both the ordering of the policy objects and the ordering of 'policy' attributes within policy objects, if the long format of the policy class is used. The physical ordering is the ordering of the policies in a file of SPSL policy objects. This ordering must be maintained by the parser and other applications that use the SPSL objects.

6. Security Considerations

SPSL is used to define a set of security policies for a host or a domain. It is necessary to insure that the policies are only modified by authorized maintainers, so that the intended policies are enforced. The language provides the mechanisms to insure this and the integrity of the policies, however the mechanisms must be used to secure them.

Tools that create and modify SPSL objects MUST use the 'auth' attribute in the mntner object to authenticate the maintainer before permitting any objects to be modified. When defining the maintianer initially, the relative strengths of the provided authentication mechanisms SHOULD be considered before using a particular one. The integrity of an SPSL policy file is only as strong as the weakest 'auth' mechanism provided.

Tools that modify or use SPSL objects SHOULD verify the signatures on the objects before using them. A successful verification indicates that the policy was written or modified by an authorized maintainer. If the policy fails verification, it is suspect and SHOULD NOT be used.

7. Remaining Issues

The following issues are not resolved in this first draft of language definition. Solutions will be developed and included in the subsequent revisions of the document.

* General SA endpoints need to be thought out more, as noted in the document.

* We are considering adding support for DNS names as policy endpoints and for domain coverage in addition to IP addresses.

8. Acknowledgements

The authors thank Luis Sanchez, David Mankins, Alden Jackson, and Steve Kent for their help in reviewing early drafts of this document and suggesting changes to the language. We thank Rajesh Krishnan and Matt Fredette for their work on an SPSL parser and suggested changes to make the language parseable. We also thank Pam Helinek and Marla Shepard for building an interface for creating, modifying, and processing SPSL files.

Appendix A. BNF Form of SPSL

```
   spsl-file -> spslobjlist | (empty)
   spslobjlist -> spslobjlist spslobj | spslobj
   spslobj -> "mntner:" objectname line-term mntner-attributes blankline
        | "cert:" objectname line-term cert-attributes blankline
        | "node:" objectname line-term node-attributes blankline
        | "node-set:" objectname line-term node-set-attributes blankline
        | "gateway:" objectname line-term gateway-attributes blankline
        | "gateway-set:" objectname line-term gateway-set-attributes
              blankline
        | "domain:" objectname line-term domain-attributes blankline
        | "polserv:" objectname line-term polserv-attributes blankline
        | "policy-name:" objectname line-term policy-attributes blankline
        | line-term

   # checking for mandatory attributes is necessary after parsing
   mntner-attributes -> mntner-attributes mntner-attribute
        | mntner-attribute
   cert-attributes -> cert-attributes cert-attribute | cert-attribute
   node-attributes -> node-attributes node-attribute | node-attribute
   node-set-attributes -> node-set-attributes node-set-attribute
        | node-set-attribute
   gateway-attributes -> gateway-attributes gateway-attribute
        | gateway-attribute
   gateway-set-attributes -> gateway-set-attributes gateway-set-attribute
        | gateway-set-attribute
   domain-attributes -> domain-attributes domain-attribute
        | domain-attribute
   polserv-attributes -> polserv-attributes polserv-attribute
        | polserv-attribute
   policy-attributes -> policy-attributes policy-attribute
        | policy-attribute

   mntner-attribute -> shared-attribute | "auth:" auth-info line-term
        | "address:" string line-term | "phone:" phonenum line-term
        | "fax-no:" phonenum line-term | "email:" emailaddr line-term
        | "certs:" objectnamelist line-term

   cert-attribute -> shared-attribute
        | "certificate:" certtype hexstring line-term
        | "certlocation:" certtype fetchproto locname line-term
        | "crllocation:" crltype fetchproto locname line-term

   node-attribute -> shared-attribute | "name:" dnsname line-term
        | "alias:" dnsname line-term | "ifaddr:" ipaddress line-term

   node-set-attribute -> shared-attribute
        | "members:" objectnamelist line-term

   gateway-attribute -> shared-attribute | "name:" dnsname line-term
        | "alias:" dnsname line-term | "ifaddr:" ipaddress line-term
        | "preference:" integer line-term
```

```
   gateway-set-attribute -> shared-attribute
        | "members:" objectnamelist line-term

   domain-attribute -> shared-attribute
        | "coverage:" domaincover line-term
        | "gateways:" objectnamelist line-term
        | "polservs:" objectnamelist line-term

   polserv-attribute -> shared-attribute | "name:" dnsname line-term
        | "alias:" dnsname line-term | "ifaddr:" ipaddress line-term

   policy-attribute -> shared-attribute
        | "association:" objectnamelist line-term
        | "cache-expiry:" integer line-term | condition-attribute
        | action-attribute

   shared-attribute -> "char-set:" charset line-term
        | "notes:" string line-term | "mnt-by:" objectnamelist line-term
        | "changed:" objectname date line-term
        | "signature:" objectname objectname hash-alg signature-data
              line-term
        | comments blankline

   condition-attribute -> "policy:" "dst" addresslist ports-opt src-opt
              xport-opt dir-opt actiontype line-term
        | "valid-period:" valid-period-list line-term
        | "dst:" addresslist ports-opt line-term
        | "src:" addresslist ports-opt line-term
        | "xport-proto:" integerlist line-term
        | "direction:" dirtype symmetric-opt line-term
        | "userid:" user-namelist line-term
        | "systemname:" system-namelist line-term
        | "ipv6-class:" integerlist line-term
        | "ipv6-flow:" integerlist line-term
        | "ipv4-tos:" integerlist line-term
        | "seclabel:" seclabellist line-term
        | "ipver:" integerlist line-term
        | "ipv4-hlen:" integerlist line-term
        | "ipv4-tlen:" integerlist line-term
        | "ipv4-id:" integerlist line-term
        | "ipv4-frgm:" zeroone line-term
        | "ipv4-frgo:" integerlist line-term
        | "ipv4-ttl:" integerlist line-term
        | "ipv4-prot:" integerlist line-term
        | "ipv4-src:" ipv4list line-term
        | "ipv4-dst:" ipv4list line-term
        | "ipv4-opt-lsrr-dst:" ipv4list line-term
        | "ipv4-opt-ssrr-dst:" ipv4list line-term
        | "ipv6-dst:" ipv6list line-term
        | "ipv6-src:" ipv6list line-term
        | "ipv6-nhdr:" integerlist line-term
        | "rh-nhdr:" integerlist line-term
        | "rh-vers:" integerlist line-term
```

```
        | "rhv1-dst:" ipv6list line-term
        | "rhv1-nhop:" ipv6list line-term
        | "rhv1-phop:" ipv6list line-term
        | "ah-nhdr:" integerlist line-term
        | "dop-nhdr:" integerlist line-term
        | "esp-nhdr:" integerlist line-term
        | "frag-nhdr:" integerlist line-term
        | "hop-nhdr:" integerlist line-term
        | "ipcomp-nhdr:" integerlist line-term
        | "tcp-ack:" zeroone line-term
        | "tcp-dato:" integerlist line-term
        | "tcp-dst-port:" integerlist line-term
        | "tcp-fin:" zeroone line-term
        | "tcp-psh:" zeroone line-term
        | "tcp-rst:" zeroone line-term
        | "tcp-src-port:" integerlist line-term
        | "tcp-syn:" zeroone line-term
        | "tcp-urg:" zeroone line-term
        | "tcp-urgp:" integerlist line-term
        | "udp-cks:" integerlist line-term
        | "udp-dst-port:" integerlist line-term
        | "udp-src-port:" integerlist line-term
        | "icmp4-code:" integerlist line-term
        | "icmp4-gwy:" ipv4list line-term
        | "icmp4-id:" integerlist line-term
        | "icmp4-mtu:" integerlist line-term
        | "icmp4-seq:" integerlist line-term
        | "icmp4-type:" integerlist line-term
        | "icmp6-code:" integerlist line-term
        | "icmp6-gwy:" ipv6list line-term
        | "icmp6-id:" integerlist line-term
        | "icmp6-mtu:" integerlist line-term
        | "icmp6-seq:" integerlist line-term
        | "icmp6-type:" integerlist line-term

   action-attribute -> "tfr-action:" actiontype line-term
        | ipsec-attribute
   actiontype -> actionpd | actionfwd | actionpd "," actionfwd
   actionpd -> "permit" | "deny"
   actionfwd -> "forward" actionfwd-dst
   actionfwd-dst -> "dns" dnsname | ipaddress

   addresslist -> ipaddrlist | "any" | "*"

   auth-info -> "crypt-pw" string | "pgp" hexstring
        | "cert" objectnamelist

   certtype -> "dnskey" | "kerberos" | "pgp" | "pkcs7" | "spki"
        | "x509_ke" | "x509_sig"
   crltype -> "x509"

   date -> digit digit digit digit digit digit digit digit
```

```
   dir-opt -> "direction" dirtype symmetric-opt | (empty)
   dirtype -> "inbound" | "outbound"
   symmetric-opt -> "," "symmetric" | (empty)

   dn -> # see rfc 1779

   # DNS name based on RFC 1034
   dnsnamelist -> dnsnamelist "," dnsnamecomp
   dnsnamecomp -> dnsname
   dnsname -> dnsname "." label | label
   label -> letter label-end-opt
   label-end-opt -> ldh-string letdig | (empty)
   ldh-string -> ldh-string letdighyph | (empty)
   letdighyph -> letter | digit | "-"
   letdig -> letter | digit

   domaincover -> ipaddrlist | objectnamelist
        | ipaddrlist "," objectnamelist

   edipn -> string

   # email address from rfc 822
   emailaddr -> username "@" dnsname

   expiry-opt -> expiry | (empty)
   expiry -> "expiry" expiry-type integerrange
   expiry-type -> "seconds" | "kilobytes"

   fetchproto -> "cdp" | "dns" | "file"
   field-opt -> "field" integer | (empty)
   filename -> filename filechar | (empty)
   filechar -> alphanum | "_" | "-" | ":" | "." | "/" | "\"

   genname -> "dirname" rdnlist | "dns" dnsname | "ediname" edipn
        | "ipaddr" ipaddress | "n822" emailaddr | "other" oid string
        | "regid" oid | "uri" uri | "x400" or-address

   hash-alg -> "dsa-sha1" | "rsa-pkcs1"

   integerlist -> integerslist | "any" | "opaque" | "*"
   integerslist -> integerslist "," integercomp | integercomp
   integercomp -> integerrange | "not" integerrange
   integerrange -> "min" integer | "max" integer
        | integer "-" integer | integer
   integer -> integer digit | digit

   ipaddress -> ipv4address | ipv6address
   ipv4address -> two55 "." two55 "." two55 "." two55
   ipv6address -> v6digit ":" v6digit ":" v6digit ":" v6digit ":"
        v6digit ":" v6digit ":" v6digit ":" v6digit
   v6digit -> hexdigit | hexdigit hexdigit | hexdigit hexdigit hexdigit
        | hexdigit hexdigit hexdigit hexdigit | (empty)
```

```
   ipaddrlist -> ipaddrlist "," ipcomp | ipcomp
   ipcomp -> ipv4comp | ipv6comp

   ipv4list -> ipv4list "," ipv4comp | ipv4comp
   ipv4comp -> ipv4address | ipv4address-range
        | "not" ipv4address | "not" ipv4address-range
   ipv4address-range -> ipv4address "-" ipv4address
        | ipv4address "mask" ipv4address
        | ipv4address "/" integer

   ipv6list -> ipv6list "," ipv6comp | ipv6comp

   ipv6comp -> ipv6address | ipv6address-range
        | "not" ipv6address | "not" ipv6address-range
   ipv6address-range -> ipv6address "-" ipv6address
        | ipv6address "mask" ipv6address
        | ipv6address "/" integer

   ipsec-attribute -> "ipsec-action:"  ipsec-action line-term
        | "ike-action:" ike-action line-term

   ipsec-action -> ipsec_action_esp_opt ipsec_action_ah_opt
        ipsec_action_ipcomp_opt

   ipsec_action_esp_opt -> esp-proposal ipsectype ipsecloc | (empty)
   ipsec_action_ah_opt -> ah-proposal ipsectype ipsecloc | (empty)
   ipsec_action_ipcomp_opt -> ipcomp-proposal | (empty)

   ipsectype -> "tunnel" | "transport" | (empty)
   usepfs -> "true" | "false"

   proposal-choice -> "req" | "opt"

   ah-proposal -> "ah" proposal-choice "integrity"
        integrity-alg-any expiry-opt | "ah" "proh"
   integrity-alg-any -> "any" keylen-opt | integrity-alg-list
   integrity-alg-list -> integrity-alg-list "," not-opt integrity-alg
        keylen-opt | not-opt integrity-alg keylen-opt
   integrity-alg -> "hmacdes" | "hmacmd5" | "hmacsha1"
        | "kpdk" | integer

   esp-proposal -> "esp" proposal-choice "cipher" ipsec-cipher-alg-any
        integrity-opt expiry-opt | "esp" "proh"
   ipsec-cipher-alg-any -> "any" keylen-opt rounds-opt
        | ipsec-cipher-alg-list
   ipsec-cipher-alg-list ->
        ipsec-cipher-alg-list "," not-opt ipsec-cipher-alg keylen-opt
        rounds-opt | not-opt ipsec-cipher-alg keylen-opt rounds-opt
   ipsec-cipher-alg -> "blowfish" | "cast" | "des" | "des3" | "idea"
        | "idea3" | "none" | "rc4" | "rc5" | "rfc1829-iv32"
        | "rfc1829-iv64" | integer
   rounds-opt -> "rounds" integerrange | (empty)
   integrity-opt -> "integrity" integrity-alg-any | (empty)
```

```
   ipcomp-proposal -> "ipcomp" proposal-choice ipcomp-alg-any
        | "ipcomp" "proh"
   ipcomp-alg-any -> "any" | ipcomp-alg-list
   ipcomp-alg-list -> ipcomp-alg-list "," not-opt ipcomp-alg
        | not-opt ipcomp-alg
   ipcomp-alg -> "deflate" | "lzs" | "oui" | integer

   ike-action -> "ikemode" ikemode "pfs" usepfs
        "auth" ike-auth-method-any
        "cipher" ike-cipher-alg-any "hash" ike-hash-alg-any
        ike-group-opt prf-opt field-opt expiry

   ikemode -> "main" | "aggressive" | "quick"

   ike-auth-method-any -> "any" | ike-auth-method-list
   ike-auth-method-list -> ike-auth-method-list "," not-opt
        ike-auth-method | not-opt ike-auth-method
   ike-auth-method -> "pre-shared" | "dss" | "rsa" | "rsa-encrypt"
        | "rsa-revised" | integer

   ike-cipher-alg-any -> "any" keylen-opt | ike-cipher-alg-list
   ike-cipher-alg-list -> ike-cipher-alg-list "," not-opt ike-cipher-alg
        keylen-opt | not-opt ike-cipher-alg keylen-opt
   ike-cipher-alg -> "blowfish" | "cast" | "des" | "des3" | "idea"
        | "rc5" | integer

   ike-hash-alg-any -> "any" | ike-hash-alg-list
   ike-hash-alg-list -> ike-hash-alg-list "," not-opt ike-hash-alg
        | not-opt ike-hash-alg
   ike-hash-alg -> "md5" | "sha1" | "tiger" | integer

   ike-group-opt -> "group-desc" ike-group-desc |
                    "group-type" ike-group-type | (empty)
   ike-group-desc -> "modp-768" | "modp-1024" | "ec2n-155"
        | "ec2n-185" | integer
   ike-group-type -> ike-group-name hexstring hexstring hexstring
        hexstring hexstring hexstring
   ike-group-name -> "modp" | "ecp" | "ec2n" | integer

   ipsecloc -> from-opt to-opt
   from-opt -> "from" anylocation | (empty)
   to-opt -> "to" anylocation | (empty)
   anylocation -> "any" | locations
   locations -> locations "," location | location
   location -> "dest" | "host" | "local-sg" | "remote-sg" | ipaddress
        | "dns" dnsname

   keylen-opt -> "keylen" integerrange | (empty)
   locname -> genname | "rdn" rdn | "filename" filename
   not-opt -> "not" | (empty)
```

```
   objectnamelist -> objectnamelist "," objectname | objectname
   objectname -> extletter objectinternals alphanum | extletter alphanum
        | extletter
   objectinternals -> objectinternals objectinternal | (empty)
   objectinternal -> alphanum | "_" | "-" | ":" | "."

   oid -> objectname
   or-address -> string
   phonenum -> phonenum phonenum | digit | " " | "+" | "-" | "x"

   ports-opt -> "port" integerlist dynamic-opt | (empty)
   dynamic-opt -> "dynamic" portrange-opt | (empty)
   portrange-opt -> integerrange | (empty)

   prf-opt -> "prf" integer | (empty)
   rdnlist -> rdnlist rdn | rdn
   rdn -> # see rfc 1779

   src-opt -> "src" addresslist ports-opt | (empty)
   seclabellist -> seclabelslist | "*" | "opaque"
   seclabelslist -> seclabelslist "," not-opt seclabel | not-opt seclabel
   seclabel -> hexstring
   signature-data -> hexstring

   system-namelist -> system-nameslist | "*" | "any"
   system-nameslist -> system-nameslist "," not-opt system-name
        | not-opt system-name
   system-name -> genname | "dn" dn

   uri -> # see appendix A of draft-fielding-uri-syntax-03.txt
   username -> # see definition in RFC 822

   user-namelist -> user-nameslist | "*" | "any"
   user-nameslist -> user-nameslist "," not-opt user-name
        | not-opt user-name
   user-name -> "n822" emailaddr | "dn" dn

   valid-period-list -> valid-period-list "," valid-period
        | valid-period
   valid-period -> year-opt month-opt dayof-month-opt
        dayof-week-opt time-opt

   year-opt -> "year" integer "-" integer | (empty)
   month-opt -> "month" bitstring | (empty)
   dayof-month-opt -> "day-of-month" bitstring | (empty)
   dayof-week-opt -> "day-of-week" bitstring | (empty)
   time-opt -> "time" not-opt time "-" time | (empty)
   time -> integer ":" integer ":" integer

   xport-opt -> "xport-proto" integerlist | (empty)
```

```
   alphanum -> extletter | digit
   charset -> string
   digit -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
   extletter -> A..Z | a..z | #140 | #156 | #192..#214
        | #216..#246 | #248..#255
   letter -> A..Z | a..z
   two55 -> [0-9] | [0-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5]
   hexstring -> hexstring hexdigit | hexdigit
   hexdigit -> [0-9] | a | A | b | B | c | C | d | D | e | E | f | F
   string -> string char | (empty)
   zeroone -> 0 | 1
   bitstring -> bitstring zeroone | zeroone

   line-term -> comments blankline | blankline
   comments -> comments comment | comment
   comment -> "#" string

   blankline -> whitespace LF
   whitespace -> whitespace whitechar | (empty)
   whitechar -> tab | " " | ff

   char -> any character in ISO 8859-1 (Latin-1) except special characters
        ("#" and "\") which must be replaced by their escaped versions
        ("\#" and "\\").
```

References

[Bra97] S. Bradner, "Key words for use in RFCs to Indicate Requirement Level," RFC-2119, March 1997.

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

[DSA] Federal Information Processing Standards Publication (FIPS PUB) 186, Digital Signature Standard, 18 May 1994.

[ISO8859] Information Processing - 8-bit Single-Byte Coded Graphic Character Sets. Part1: Latin Alphabet Number 1, ISO 8859-1, 1987.

[Kent98] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[PKIXP1] R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 2459, January 1999.

[PolMod] R. Pereira, P. Bhattacharya, "IPSec Policy Data Model", Internet Draft draft-ietf-ipsec-policy-model-00, February 1998.

[rfc822] D. Crocker, "Standard for the Format of ARPA Internet Text Messages", RFC 822, August 1982.

[rfc1034] P. Mockapetris, "Domain Names - Concepts and Facilities", RFC 1034, November 1987.

[rfc2409] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RPSL] C. Alaettinouglu, T. Bates, E. Gerich, D. Karrenberg, D. Meyer, M. Terpstra, and C. Villamizer. "Routing Policy Specification Language (RPSL)". RFC 2280. January 1998.

[RSA] PKCS #1: RSA Encryption Standard, Version 1.4, RSA Data Security, Inc., 3 June 1991.

[SPS] L. Sanchez, M. Condell, "Security Policy System", Internet Draft draft-ietf-ipsec-sps-00.txt, November 1998.

Author Information

Matthew Condell
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
USA
Email: mcondell@bbn.com
Telephone: +1 (617) 873-6203

Charles Lynn
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
USA
Email: clynn@bbn.com
Telephone: +1 (617) 873-3367

John Zao
BBN Technologies
10 Fawcett Street
Cambridge, MA 02138
USA
Email: jzao@bbn.com
Telephone: +1 (617) 873-2438