Session Initiation Proposal
Investigation Working Group
Internet-Draft
Expires: April 24, 2005

V. Hilt
Bell Labs/Lucent Technologies
G. Camarillo
Ericsson
J. Rosenberg
Cisco Systems

October 24, 2004

Session-Independent Session Initiation Protocol (SIP) Policies - Mechanism and Policy Schema
draft-ietf-sipping-session-indep-policy-01

Status of this Memo

Copyright Notice

Abstract

This draft specifies an XML schema for profile data for SIP session-policies. It defines a delivery mechanism for SIP session-policies that are independent of a specific SIP session.

Table of Contents

1. Introduction

Some domains have policies in place, which impact the sessions established using the Session Initiation Protocol (SIP) [10]. These policies are often needed to support the network infrastructure or the execution of services. For example, wireless networks usually have limited resources for media traffic. During periods of high activity, a wireless network provider may want to restrict codec usage on the network to lower rate codecs.

In another example, a SIP user agent is using a network which connects to the public Internet through a firewall at the network edge. The provider would like to tell the user agent to direct the media streams to the appropriate open ip/ports on that firewall. Knowing this policy enables these user agents to setup sessions with user agents on the public Internet.

In a third example, a domain A has a limited bandwidth pipe to another domain B. The pipe is realized through two routers that are managed. Domain A would like to direct each call to one of the routers based on their capacity. With session policies, the domain can inform the user agent about the route with the most capacity available.

Domains sometimes enforce policies they have in place. For example, a domain might have a configuration in which all packets containing a certain audio codec are dropped. Unfortunately, enforcement mechanisms usually do not inform the user about the policies they are enforcing and silently keep the user from doing anything against them. This may lead to the malfunctioning of devices that is incomprehensible to the user. With session policies, the user knows about the restricted codecs and can use a different codec or simply connect to a domain with less stringent policies. Session policies provide an important combination of consent coupled with enforcement. That is, the user becomes aware of the policy and needs to act on it, but the provider still retains the right to enforce the policy.

Some policies are created specifically for a certain session. Such policies usually require that the user agent provides information about the session to the network and receives policies that are tailored to this session. Session-specific policies can be set up using the framework for session-specific policies [3].

Session policies are often independent of a certain session and generally apply to the sessions that are set up by a user agent. In principle, these policies could also be set up on a session-to-session basis. However, it is much more efficient to enable user agents to obtain the policies relevant for them and to

inform the user agents about changes in these policies. This draft defines a mechanism for delivering session-independent policies to user agents.

This draft also defines XML schemas for session policies. It defines a general session policy schema acting as a framework for session policies. It also defines schemas for media policies, protocol policies and media routing policies. Policy instance documents can be delivered to user agents as session-independent policies using the mechanisms below or as session-specific policies [3].

Note: The difference between session-independent and session-specific policies needs more discussion here.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, [1] and indicate requirement levels for compliant implementations.

3. Session-Independent Policy Mechanism

This draft defines a mechanism for the delivery of session-independent policies to UAs. Session-independent policies can be part of the device configuration and reside on the same configuration server. They can be delivered to UAs in conjunction with the device configuration. Session-independent policies may also be independent of other configuration information and reside on different servers.

This mechanism enables UAs to indicate their support for session policies and to receive policies from different policy servers. A UA subscribes to session-independent policies. It receives these policies when the subscription is established and is notified if updates to session policies become available. The session-independent policy mechanism is based on the Framework for SIP User Agent Profile Delivery [7] and RFC3265 [9].

3.1 Subscriber Behavior

A UA can express interest in session-independent policies by subscribing to session policies a policy server. If the UA has already received the URIs of all relevant session policy documents (e.g., through configuration) it SHOULD use these URIs to create a subscription as defined in [7].

Session-independent policies are frequently provided to a UA by the following two network domains: the domain a user registers at (i.e., the domain in the address-of-record (AoR)) and the domain the UA is physically connected to. The AoR-domain may, for example, provide policies that are needed for services the user has subscribed. The domain that provides the physical network connection may hand have policies helping to maintain the functionality of the network, e.g., by limiting the bandwidth a single UA can consume. A UA can subscribe to these two domains without having previous knowledge about the policy server URI by using the profile-types "user" and "local" defined in [7]. Since a UA has no way of knowing whether a domain has a policy server, it SHOULD attempt to subscribe to these two profile-types if the following events occur:

- One (or more) AoRs registered by the UA change. This might be due to a change in the AoR of an existing user or a user is added to/removed from the set of users of a device. The UA SHOULD subscribe each AoR that as changed using the "user" as well as the "local" profile-type. It SHOULD terminate all subscriptions for AoRs not in use any more.
- The domain the UA is connected to changes. The UA SHOULD create a subscription for each AoR it maintains using the "local" profile-type. It SHOULD terminate all existing subscriptions for the "local" profile-type.

If a subscriber is unable to successfully establish a subscription, it SHOULD NOT attempt to re-try this particular subscription at a later point, unless one of the above events occurs again. This is to limit the number of SUBSCRIBE requests sent within domains that do not support session-policies.

A subscriber compliant to this specification MUST indicate its support for session policies by adding the MIME types of supported session policy formats to the Accept header of the SUBSCRIBE request. This specification defines the new MIME type "application/session-policy+xml". All UAs compliant to this specification MUST support this MIME type and MUST include it in the Accept header of SUBSCRIBE requests.

OPEN ISSUE: The subscriber also needs to be able to indicate support for a certain XML schema, i.e., an XML namespace!

A subscriber may receive a 406 in response to a SUBSCRIBE request.

This indicates that the notifier requires the support of a session policy format that was not in the Accept header of the SUBSCRIBE request. This means that the notifier has session policies that are required in the network but not supported by the subscriber. As a consequence, the subscriber may experience difficulties when setting

up a session without these policies.

3.2 Notifier Behavior

A network may have session policies in place that must be supported by a UA. UAs that do not support these policies may experience problems when setting up a session. For example, a network may have a policy enabling firewall traversal.

A UA lists all the session policy formats it supports in the Accept header of a SUBSCRIBE request. If the notifier receives a SUBSCRIBE request that does not contain the MIME types of all policy formats required in the network, it MUST reject the request with a 406 response. A policy format is required, if the network has policy documents in this format and these documents contain constraints that must be applied by the UA. The notifier SHOULD NOT return a 406 if an unsupported format only contains optional policies.

4. Session Policy Schemas

The schemas for session policies defined in this document extend the schema for user agent profile data sets [8]. This simplifies the processing of policy data and other user agent configuration data and enables them to share the delivery mechanisms and to co-exist in the same instance documents.

This draft specifies a general schema for session policies, which covers the common aspects of session policies. It acts as a framework schema for session policies. Based on this framework, specific session policy schemas can be defined. This draft defines policy schemas for media policies, protocol policies and media routing policies. It is expected that other session policy schemas will be defined in the future.

This specification makes use of XML namespaces. The namespace URIs for schemas defined in this specification are URNs [5], using the namespace identifier 'ietf' defined by RFC 2648 [6] and extended by [4].

4.1 Specifying Constraints

Policies are defined by using the constraint elements defined in [8]. The specification of constraints is centered around the concept of a working profile. A working profile is the set of properties a UA is actually using during operation. The following sections defines how session policies constraints influence the working profile of a UA.

forbid - the values of elements contained inside of a forbid element MUST NOT be used in the working profile. If a policy element value in a forbid element is in the working profile, it MUST be removed. For example, the forbid element may contain the name of codecs that are prohibited in the local local network. The forbid element can contain empty policy elements. This means that all possible values for that element MUST be removed from the working profile. For example, a <codec /> element in the forbid container indicates that all codecs must be removed from the working profile. Specific codecs can be added to the working profile again by listing them in a set_all, set_one or set_any element inside the same the same property_set. Policy element values that were removed in one property_set can't be added again by a different property_set. This would constitute a policy conflict between the two property_sets.

set_all - the set_all element contains policy element values that MUST be present in the working profile of a UA. They MUST be added to the working profile if they are supported by the UA and not forbidden by another property_set. A policy conflict occurs if they can't be added to the working profile.

set_one - the semantics of the set_one element is similar to the set_all element except that the set_one element may contain alternative policy element and the UA MUST apply the first policy element that does not cause conflicts.

set_any - the set_any element contains policy element values that SHOULD be added to the working profile if they are supported by the UA and do not conflict with other policies.

A UA MUST process the forbid element before processing the set_all, set_one, and set_any elements within a property_set.

Note: The structure of the schema and the way constraints are specified has changed significantly since the last revision. The goal was to better align with the profile data set framework and to simplify the specification of policies.

4.2 Extensibility

Other elements from different namespaces MAY be present within an element of a policy schema for the purposes of extensibility; elements or attributes from unknown namespaces MUST be ignored.

4.3 General Policy Schema

The general session policy schema defines the elements and attributes that are common to all session policy schemas. All session policy documents MUST import this schema.

The namespace of the general session policy schema is:
urn:ietf:params:xml:ns:sessionpolicy

This document specifies a MIME type for policy instance documents. The new MIME type is:
application/session-policy+xml

4.3.1 Elements and Attributes

The following elements are defined in the session policy schema. They are optional elements that MAY appear once inside a settings element [8]. They do not refer to a particular property set and therefore appear outside of a property_set element.

4.3.1.1 Version

The version element allows the recipient to properly order setting instances. Versions start at 0, and increment by one for each new document sent to a subscriber. Versions are scoped within a subscription. Versions MUST be representable using a 32 bit integer.

4.3.1.2 Domain

The domain element contains a URI that identifies the domain to which this setting instance belongs to.

4.3.1.3 Entity

The entity element contains a URI that identifies the user or device whose policy information is reported in this setting instance. If this element is not present, the setting applies to all entities on a device.

4.3.1.4 Include Target

The includeTarget element contains a URI that identifies the remote target of a dialog. A setting which in which this element is contained applies to all dialogs which have this URI as their remote

target. Missing URI elements MUST match to any value. For example, a URI without a user name matches to all users in this domain.

4.3.1.5 Exclude Target

The excludeTarget element contains a URI that identifies the remote target of a dialog. The setting in which this element is contained applies to all dialogs which do not have this URI as the remote target. Missing URI elements MUST match to any value. For example, a URI without a user name matches to all users in this domain.

4.3.1.6 Info

The info element provides a short textual description of the setting that should be intelligible to the human user.

4.4 Media Policy Schema

The media policy schema defines the elements and attributes needed to describe the characteristics of media streams.

The namespace of the media policy schema is:
urn:ietf:params:xml:ns:mediapolicy

4.4.1 Elements and Attributes

The following elements and attributes are defined in the media policy schema.

4.4.1.1 Media

The media element defines a media policy. A media element MAY appear zero, one or multiple times in a constraint element.

The media element has the following sub-elements.

4.4.1.1.1 Max Bandwidth

The maxBandwidth element defines the maximum bandwidth in kilobits per second an entity can count on.

The maxBandwidth element is optional and can only appear once within a media element. All subsequent instances MUST be ignored. It has no meaning and MUST be ignored if it is inside a forbid constraint.

4.4.1.1.2 Maxno Streams

The maxnoStreams element defines the maximum number of streams an entity is allowed to handle at the same time.

The maxnoStreams element is optional and can only appear once within a media element. All subsequent instances MUST be ignored. It has no meaning and MUST be ignored if it is inside a forbid constraint.

4.4.1.1.3 Maxno Sessions

The maxnoSessions element defines the maximum number of sessions an entity is allowed to establish at the same time.

The maxnoSessions element is optional and can only appear once within a media element. All subsequent instances MUST be ignored. It has no meaning and MUST be ignored if it is inside a forbid constraint.

4.4.1.1.4 Maxno Streams Session

The maxnoStreamsSession element defines the maximum number of streams an entity is allowed to establish within a session.

The maxnoStreamsSession element is optional and can only appear once within a media element. All subsequent instances MUST be ignored. It has no meaning and MUST be ignored if it is inside a forbid constraint.

4.4.1.1.5 Max Bandwidth Session

The maxBandwidthSession element defines the maximum bandwidth in kilobits per second available to the entity within one session.

The maxBandwidthSession element is optional and can only appear once within a media element. All subsequent instances MUST be ignored. It has no meaning and MUST be ignored if it is inside a forbid constraint.

4.4.1.1.6 Max Bandwidth Stream

The maxBandwidthStream element defines the maximum bandwidth in kilobits per second available to the entity for one stream.

The maxBandwidthStream element is optional and can only appear once within a media element. All subsequent instances MUST be ignored. It has no meaning and MUST be ignored if it is inside a forbid constraint.

4.4.1.1.7 Type

The type element identifies a media type. The value of this element MUST be the name of a IANA registered media type (see [2]), such as

"audio", "video", "text", or "application".

The type element is optional and MAY appear multiple times within a media element.

4.4.1.1.8 Codec

The codec element identifies a media codec. The value of this element MUST be the name of a registered MIME type for a encoding (see [2]), such as "PCMA", "G729", or "H263".

The codec element is optional and MAY appear multiple times within a media element.

4.4.1.1.9 Transport

The transport element identifies a transport protocol for media streams. The value of this element MUST be the name of a IANA registered protocol (see [2]), such as "udp", "RTP/AVP", or "RTP/SAVP".

The transport element has an optional attribute:

type - the type attribute identifies the media type to which the transport element applies to. The value of this attribute MUST be a legal type element value.

The transport element is optional and MAY appear multiple times within a media element.

4.4.1.1.10 Direction

The direction element identifies a media stream direction. The value of this element MUST be "sendrec", "sendonly", or "recvonly".

The direction element has an optional attribute:

type - the type attribute identifies the media type to which the direction element applies to. The value of this attribute MUST be a legal type element value.

The direction element is optional and MAY appear multiple times within a media element.

4.4.2 Conflict Resolution

The UA SHOULD alert the user in case a media policy conflicts with another policy.

OPEN ISSUE: Can we be more specific what to do if a conflict occurs in the general case?

4.4.3 Example

The following example describes a policy that limits the number of streams a UA can create to 4. It only allows audio streams and prohibits the use of the PCMU and PCMA codecs. It requires the UA to support RTP/AVP as a transport and optionally allows RTP/SAVP but no other transport protocols. Audio streams can only be sent.

```
<property_set>
  <forbid>
    <media>
      <type />
      <codec>PCMU</codec>
      <codec>PCMA</codec>
      <transport />
    </media>
  </forbid>
  <set_all>
    <media>
      <maxnoStreams>4</maxnoStreams>
      <type>audio</type>
      <transport>RTP/AVP</transport>
      <direction type="audio">sendonly</direction>
    </media>
  </set_all>
  <set_any>
    <media>
      <transport>RTP/SAVP</transport>
    </media>
  </set_any>
</property_set>
```

4.5 Protocol Policy Schema

The protocol policy schema defines the elements and attributes needed to describe protocol characteristics.

The namespace of the protocol policy schema is:
urn:ietf:params:xml:ns:protocolpolicy

4.5.1 Elements and Attributes

The following elements and attributes are defined in the protocol policy schema.

4.5.1.1 Protocol

The protocol element defines a protocol policy. Each protocols

element contains the policy related to the usage of a particular protocol. A protocol element MAY appear zero, one or multiple times in a constraint element.

The protocol element has an optional attribute:

name - the name attribute identifies the name of the protocol, to which the policy of the protocol element is referring to.

Each protocol element has the following sub-elements.

4.5.1.1.1 Proxy

The proxy element identifies the URI of a proxy. The proxy values MUST be used to create a route set.

The proxy element is optional and may appear multiple times inside a protocol element. Multiple appearances of this element are ordered. It has no meaning and MUST be ignored if it is inside a forbid constraint.

4.5.1.1.2 Method

The method element identifies a method. The value of this element MUST be the name of a valid method within the protocol identified by in the protocol element.

The method element is optional and MAY appear multiple times within a protocol element.

4.5.1.1.3 Option Tag

The optionTag element identifies an optionTag. The value of this element MUST be the name of an option tag registered for the protocol identified by in the protocol element in the IANA registry.

The optionTag element is optional and MAY appear multiple times within a protocol element.

4.5.1.1.4 Transport

The transport element identifies a transport protocol for the protocol identified by the protocol element. The value of this

element MUST identify a valid transport for the given protocol. For SIP, the value MUST a value that is registered for the transport header field parameter, such as "TCP", "UDP", or "SCTP".

The transport element is optional and MAY appear multiple times within a protocol element.

4.5.1.1.5 Body-disposition

The body-disposition element identifies a body-disposition. The value of this element MUST be a name registered in the IANA registry

for Content-Dispositions.

The body-disposition element is optional and MAY appear multiple times within a protocol element.

4.5.1.1.6 Body-format Element

A body-format element identifies a body-format. The value of this element MUST be the name of a MIME type registered in the IANA registry.

The body-format element has an optional attribute:

body-disposition - the body-disposition attribute identifies the body disposition used with this body-format. The value of this attribute MUST be a value legal for the body-disposition element.

The body-format element is optional and MAY appear multiple times within a protocol element.

4.5.1.1.7 Body-encryption

The body-encryption indicates whether or not encryption is allowed for a particular body disposition. This element MUST NOT have a value.

The body-encryption element has the following optional attributes:

body-disposition - the body-disposition attribute identifies the body disposition this encryption setting applies to. The value of this attribute MUST be a value legal for the body-disposition element.
body-format - the body-format attribute identifies the body format to which this encryption setting applies to. The value of this attribute MUST be a value legal for the body-format element.

The body-encryption element is optional and MAY appear multiple times within a protocol element.

4.5.2 Conflict Resolution

The UA SHOULD alert the user in case a protocol policy conflicts with another policy.

OPEN ISSUE: Can we be more specific what to do if a conflict occurs in the general case?

4.5.3 Example

The following example describes a policy saying that use of the MESSAGE method is prohibited in the network. It states that the use of the option tag 100rel is required and preconditions are allowed. Other option tags are not allowed in the network. The only MIME type for message bodies allowed is "application/sdp" with the body-disposition "session". Encryption is not allowed for bodies with body-disposition "session".

```
<property_set>
  <forbid>
    <protocol name="SIP">
      <method>MESSAGE</method>
      <optionTag />
      <body-format />
      <body-encryption body-disposition="session" />
    </protocol>
  </forbid>
  <set_all>
    <protocol name="SIP">
      <optionTag>100rel</optionTag>
      <body-format body-disposition="session">application/sdp</body-format>
    </protocol>
  </set_all>
  <set_any>
    <protocol name="SIP">
      <optionTag>preconditions</optionTag>
    </protocol>
  </set_any>
</property_set>
```

4.6 Media Routing Policy Schema

The media routing schema defines the elements and attributes needed to describe address and ports of a media stream. This schema can be used to route the media stream through a NAT, a firewall or a media relay.

The namespace of the protocol policy schema is:
urn:ietf:params:xml:ns:mediaroutingpolicy

OPEN ISSUE: This schema probably needs to go into a separate draft along with some text about the mechanics.

4.6.1 Elements and Attributes

The following elements and attributes are defined in the protocol policy schema.

4.6.1.1 Media Routing

The mediaRouting element defines a media routing policy. A media routing element MAY appear zero, one or multiple times in a constraint element.

Each protocol element has the following sub-elements.

4.6.1.1.1 Address

The address element contains the destination address of a media stream, i.e., the address that is contained in an SDP announcement for a media stream.

The address element MUST have the following attribute:

direction - the direction attribute identifies the direction of a media stream. The value of this element MUST be "send" or "recv". It determines whether the element applies to the session description offer or answer.

The address element is optional and MAY appear at most once within a media routing element.

4.6.1.1.2 Port

The port element identifies the destination port of a media stream, i.e., the address that is contained in an SDP announcement for a media stream.

The address element MUST have the following attribute:

direction - the direction attribute identifies the direction of a media stream. The value of this element MUST be "send" or "recv". It determines whether the element applies to the session description offer or answer.

The port element is optional and MAY appear multiple times within a media routing element.

4.6.1.1.3 Port Range

The portRange element identifies a range of ports that apply to the

destination of a media stream. This can, for example, be used to determine the range of ports that can be used for media streams in SDP announcements.

The address element MUST have the following attribute:

direction - the direction attribute identifies the direction of a media stream. The value of this element MUST be "send" or "recv". It determines whether the element applies to the session description offer or answer.

The portRange element is optional and MAY appear multiple times within a media routing element. It has the following two sub-elements.

4.6.1.1.3.1 Start Port

The startPort element identifies the lowest port number that belongs to the port range.

The startPort element MUST appear exactly once inside a port range element.

4.6.1.1.3.2 End Port

The endPort element identifies the highest port number that belongs to the port range.

The endPort element MUST appear exactly once inside a port range element.

4.6.2 Conflict Resolution

The UA SHOULD alert the user in case a media route policy conflicts with another policy.

OPEN ISSUE: Can we be more specific what to do if a conflict

occurs in the general case?

4.6.3 Example

The following example describes a policy that instructs the UA to use the address 123.124.125.126 and a port in the range of 8000 - 9000 in the session descriptions it creates. This information can assist a UA, for example, to traverse a NAT.

```
<property_set>
  <set_all>
    <mediaRouting>
      <address direction="recv">123.124.125.126</address>
      <portRange direction="recv">
        <startPort>8000</startPort>
        <startPort>9000</startPort>
      </portRange>
    </mediaRouting>
  </set_all>
</property_set>
```

5. Schema Definitions

5.1 General Policy Schema

[TBD.]

5.2 Media Policy Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:mediapolicy"
      xmlns:tns="urn:ietf:params:xml:ns:mediapolicy"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"
      attributeFormDefault="unqualified">


  <xs:element name="media" type="tns:media"/>


  <xs:complexType name="media">
    <xs:sequence>
      <xs:element name="maxBandwidth" type="xs:positiveInteger"
        minOccurs="0" maxOccurs="1"/>
      <xs:element name="maxnoStreams" type="xs:positiveInteger"
        minOccurs="0" maxOccurs="1"/>
      <xs:element name="maxnoSessions" type="xs:positiveInteger"
        minOccurs="0" maxOccurs="1"/>
      <xs:element name="maxnoStreamsSession"
        type="xs:positiveInteger" minOccurs="0" maxOccurs="1"/>
```

```
        <xs:element name="maxBandwidthSession"
          type="xs:positiveInteger" minOccurs="0" maxOccurs="1"/>
        <xs:element name="maxBandwidthStream"
          type="xs:positiveInteger" minOccurs="0" maxOccurs="1"/>
        <xs:element name="type" type="xs:string" minOccurs="0"
           maxOccurs="unbounded"/>
        <xs:element name="codec" type="xs:string" minOccurs="0"
```

```
            maxOccurs="unbounded"/>
          <xs:element name="transport" type="tns:transport"
            minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="direction" type="tns:direction"
            minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>

      <xs:complexType name="transport">
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="type" type="xs:string" use="optional"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>

      <xs:complexType name="direction">
        <xs:simpleContent>
          <xs:extension base="tns:directionValue">
            <xs:attribute name="type" type="xs:string" use="optional"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>

      <xs:simpleType name="directionValue">
        <xs:restriction base="xs:string">
          <xs:enumeration value="sendrec"/>
          <xs:enumeration value="sendonly"/>
          <xs:enumeration value="recvonly"/>
        </xs:restriction>
      </xs:simpleType>

    </xs:schema>
```

5.3 Protocol Policy Schema

```
    <?xml version="1.0" encoding="UTF-8"?>
    <xs:schema targetNamespace="urn:ietf:params:xml:ns:protocolpolicy"
          xmlns:tns="urn:ietf:params:xml:ns:protocolpolicy"
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">

    <xs:element name="protocol" type="tns:protocol"/>

    <xs:complexType name="protocol">
```

```
      <xs:sequence>
        <xs:element name="proxy" type="xs:anyURI" minOccurs="0"
           maxOccurs="unbounded"/>
        <xs:element name="method" type="xs:string" minOccurs="0"
           maxOccurs="unbounded"/>
        <xs:element name="optionTag" type="xs:string"
           minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="transport" type="xs:string"
           minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="body-disposition" type="xs:string"
           minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="body-format" type="tns:body-format"
           minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="body-encryption" type="tns:body-encryption"
           minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="optional"/>
    </xs:complexType>


    <xs:complexType name="body-format">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="body-disposition" type="xs:string"
            use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>


    <xs:complexType name="body-encryption">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="body-disposition" type="xs:string"
            use="optional"/>
          <xs:attribute name="body-format" type="xs:string"
            use="optional"/>
         </xs:extension>
      </xs:simpleContent>
    </xs:complexType>


  </xs:schema>
```

5.4 Media Routing Policy Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:mediaroutingpolicy"
      xmlns:tns="urn:ietf:params:xml:ns:mediaroutingpolicy"
```

```
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"
      attributeFormDefault="unqualified">

        <xs:attribute name="type" type="xs:string" use="optional"/>

  <xs:element name="mediaRouting" type="tns:mediaRouting"/>

  <xs:complexType name="mediaRouting">
    <xs:sequence>
      <xs:element name="address" type="xs:anyURI" minOccurs="0"
         maxOccurs="1"/>
      <xs:element name="port" type="tns:port"
         minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="portRange" type="tns:portRange"
         minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="port">
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="direction" type="tns:directionValue"
          use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="portRange">
    <xs:sequence>
      <xs:element name="startPort" type="xs:anyURI" minOccurs="1"
        maxOccurs="1"/>
      <xs:element name="endPort" type="xs:anyURI" minOccurs="1"
        maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="direction" type="tns:directionValue"
      use="optional"/>
  </xs:complexType>

  <xs:simpleType name="directionValue">
    <xs:restriction base="xs:string">
      <xs:enumeration value="send"/>
```

```
        <xs:enumeration value="recv"/>
      </xs:restriction>
    </xs:simpleType>


  </xs:schema>
```

6. Security Considerations

Session policy information can be sensitive information. The protocol used to distribute it SHOULD ensure privacy, message integrity and authentication. Furthermore, the protocol SHOULD provide access controls which restrict who can see who else's session policy information.

7. IANA Considerations

[TBD.]

8 References

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

[2] Handley, M., Jacobson, V. and C. Perkins, "SDP: Session Description Protocol", draft-ietf-mmusic-sdp-new-20 (work in progress), September 2004.

[3] Hilt, V., Camarillo, G. and J. Rosenberg, "A Framework for Session-Specific Session Policies in the Session Initiation Protocol (SIP)", October 2004.

[4] Mealling, M., "The IETF XML Registry", draft-mealling-iana-xmlns-registry-05 (work in progress), June 2003.

[5] Moats, R., "URN Syntax", RFC 2141, May 1997.

[6] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

[7] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-04 (work in progress), July 2004.

[8] Petrie, D., "A Schema for Session Initiation Protocol User Agent Profile Data Sets", draft-petrie-sipping-profile-datasets-00 (work in progress), July 2004.

[9] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[10] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP:

Session Initiation Protocol", RFC 3261, June 2002.

Authors' Addresses

Volker Hilt
Bell Labs/Lucent Technologies
101 Crawfords Corner Rd
Holmdel, NJ 07733
USA

EMail: volkerh@bell-labs.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
USA

EMail: jdrosen@dynamicsoft.com

Appendix A. Acknowledgements

Many thanks to Allison Mankin for the discussions and the suggestions for this draft.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.