SIPPING Working Group
Internet-Draft
Expires: January 17, 2006

V. Hilt
Bell Labs/Lucent Technologies
G. Camarillo
Ericsson
J. Rosenberg
Cisco Systems
July 16, 2005

Session Initiation Protocol (SIP) Session Policies - Document Format and Session-Independent Delivery Mechanism
draft-ietf-sipping-session-indep-policy-03

Status of this Memo

Copyright Notice

Abstract

This draft defines a document format for media-related SIP session policies. The format extends the Profile Data Set Schema by specifying a data set for media properties. This draft also defines a delivery mechanism for session policies that is independent of a

SIP session.

Table of Contents

1. Introduction

Some domains have policies in place, which impact the sessions established using the Session Initiation Protocol (SIP) [15]. These policies are often needed to support the network infrastructure or for the execution of services. For example, wireless networks usually have limited resources for media traffic. A wireless network provider may therefore restrict the bandwidth that is available to a single user. Knowing about the bandwidth limit enables an user agent to make an informed decision about the number of streams, codecs and media types it can use in a session.

In another example, a service provider wants to specifically restrict the set of codecs and media types that can be used in the network. These restrictions may change depending on network conditions. With session policies, the current set of restrictions can be conveyed to user agents to prevent them from inadvertently violating any of the network policies.

In a third example, a network provides quality of service (QoS) for media streams through differentiated services. By knowing that differentiated services are available and knowing the service class assigned to media streams, a user agent can mark the packets of media streams accordingly and therefore benefit from the QoS infrastructure.

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.

Session-policies can be set up in two different ways: specifically for a session or independent of a session. Session-specific policies are created for one particular session, usually under consideration of certain aspects of this session (e.g. the IP addresses and ports that are used for media). Since session-specific policies are tailored to a session, they only apply to the session they are created for. These policies require a delivery mechanism that enables the exchange of session policy information at the time a session is established. The framework for session-specific policies

[17] defines such a delivery mechanism for session-specific policies.

Session-independent policies on the other hand are independent of a specific session and generally apply to the sessions set up by a user agent. In principle, these policies could also be delivered to user agents individually for each session, using the session-specific policy framework. However, since these policies apply to many sessions, it is more efficient to deliver them to user agents only when the user agent is initialized or a policy changes. This draft defines a delivery mechanism for session-independent policies.

This draft also defines a document format for media-related session policies. This format is based on XML [16]. It extends the Profile Data Set Schema [13] by specifying a data set for media properties. The format defines a minimal set of media-related properties [18] and is aimed at achieving interoperability between different user agents and profile delivery/policy servers. The format can be extended through the XML extension mechanisms if additional media properties are needed. The XML document format is independent of the delivery mechanism and can be used with session-independent and session-specific session policies.

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

Session-independent policies can be delivered to UAs using the mechanism defined in the Framework for SIP User Agent Profile Delivery [12]. Session-independent policies can reside on the same server as other configuration information and they can be delivered to UAs in conjunction with this information. Session-independent policies can also reside on a separate policy server, which is independent of a configuration server. A UA may receive session-independent policies from multiple servers.

In this draft, the terms policy server and profile delivery server are used interchangeably. A policy server is a profile delivery server that provides session policies.

3.1 Subscriber Behavior

A UA can express interest in session-independent policies by

subscribing to session policies as described in [12]. If the UA already has the URIs of policy servers (e.g., through provisioning) it may directly use these URIs to subscribe to session-independent policies.

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 (i.e. the local network domain). A policy server in the AoR-domain may, for example, provide policies needed for services the user has subscribed to. The domain that provides the physical network connection may have policies needed to ensure the operativeness of the network, e.g., by limiting the bandwidth available to a UA. A UA SHOULD attempt to subscribe to the policy servers in both domains. These subscriptions are established using the "user" (for subscriptions to the AoR-domains) and the "localnetwork" (for subscriptions to the network domain) profile-types [12].

A UA SHOULD create a SUBSCRIBE request in the following events:

- The UA registers a AoR for the first time or removes a AoR from

the set of AoRs it has registered. This occurs, for example, when a UA starts up (and registers AoRs) and when it shuts down (and deregisters AoRs). This event also occurs when a new AoR is added to a UA or a AoR is removed. In these cases, the UA SHOULD establish subscriptions for each new AoR using the "user" and the "localnetwork" profile-types. It SHOULD terminate all subscriptions for the AoRs that have been removed.

- The UA changes the domain it is connected to. The UA SHOULD create a new subscription for each AoR using the "localnetwork" profile-type. It SHOULD terminate all existing subscriptions for the "localnetwork" profile-type. It does not need to change the subscriptions for "user" profiles.

If a subscriber is unable to establish a subscription, it SHOULD NOT attempt to re-try this subscription, 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 SHOULD indicate its support for session-independent 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", which MUST be supported by UAs compliant to this specification. UAs MAY also indicate support for MIME type extensions (e.g. an additional XML namespace) using [3].

4. Basic Media Policy Format

The Basic Media Policy Format (BMPF) is a document format for media-related policies. It extends the Profile Data Set Schema by providing a media data set and is used to define media-related SIP session policies.

A BMPF document is an XML [16] document that MUST be well-formed and MUST be valid according to schemas, including extension schemas, available to the validator and applicable to the XML document. BMPF documents MUST be based on XML 1.0 and MUST be encoded using UTF-8.

A user agent may receive multiple BMPF documents from different sources. These documents need to merged into a single document the user agent can work with. General rules for merging BMPF documents

are described in [13]. Specific merging rules for each of the BMPF elements are described below.

4.1 Namespace

This specification makes use of XML namespaces [4]. The namespace URIs for schemas defined in this specification are URNs [7], using the namespace identifier 'ietf' defined by [8] and extended by [5]. The namespace URN for the BMPF schema is:

urn:ietf:params:xml:ns:mediadataset

The MIME type for the Basic Media Policy Format is:

application/session-policy+xml

ISSUE: a separate MIME type might not be needed for BMPF. The MIME type of the Profile Data Set Schema may be sufficient. We still need a separate namespace.

4.2 Extensibility

The BMPF format is an extension of the Profile Data Set Schema [13]. Elements from the BMPF namespace can be used in conjunction with elements from other Profile Data Sets.

The BMPF format itself can also be extended using XML extension mechanisms. In particular, elements from different XML namespaces MAY be present within a BMPF document for the purposes of extensibility; elements or attributes from unknown namespaces MUST be ignored.

4.3 Attributes

The following attributes provide common functionalities, which are generally useful for media properties:

- Per-stream properties: 'stream-label' attribute
- Media-type specific properties: 'media-type' attribute

These attributes are defined in addition to the attributes inherited from the Profile Data Set Schema [13]:

- Property Access Control: 'visibility' attribute
- Policies: 'policy' and 'excluded-policy' attribute
- Unidirectional Properties: 'direction' attribute
- Preferences: 'q' attribute

The use of these attributes is defined individually for each element in the XML format below.

4.3.1 The 'stream-label' Attribute

Some properties only apply to a specific media stream. The stream to which a property applies to must be identifiable through a label [6]. Per-stream properties can be expressed by adding a 'stream-label' attribute to the respective element. Such a property only applies to the identified stream. If there is no stream with this label, the element must be ignored.

Per-stream properties require that the labels of media streams are known to the creator of a document (i.e. the profile delivery/policy server). These labels are, for example, part of the session description. Per-stream properties are therefore typically used for session-specific policies.

4.3.2 The 'media-type' Attribute

Some properties only apply to streams of a certain media type. For example, a property may only apply to audio streams. Media-type specific properties can be defined by adding a 'media-type' attribute to the respective element. Such a property only applies to media streams of that type.

The value of the 'media-type' attribute MUST be the name of a IANA registered media type (see [2]), such as 'audio', 'video', 'text', or 'application'.

4.4 Elements

The following elements are defined for the BMPF format.

4.4.1 The <session-policy> Element

The <session-policy> element is a container for media policy elements. It MAY occur multiple times inside a <property_set> [13] element.

The <session-policy> element MAY contain one optional <context> element and multiple (including zero) <media-types>, <codecs>, <media-intermediary>, <qos-dscp>, and <max-bandwidth> elements as well as elements from other namespaces.

> OPEN ISSUE: the <session-policy> seems to have pretty much the same functionality as the <property_set> element. Maybe it needs to be removed and the context element needs to go into the Profile Data Set Schema.

4.4.2 The <context> Element

The <context> element provides context information about this policy.

The <context> element is optional in a <session-policy> element. It MAY contain a <dialog-ID>, <domain>, multiple <contact> and an <info> element.

> Merging rule: the <context> element is not subject to merging. Information in the context element may be used to assist the user if a conflict occurs during the merging process.
> Policies that affect different sessions (i.e. have different <dialog-ID> values) are not merged.

4.4.3 The <dialog-ID> Element

Session-specific policies only apply to one particular session. The <dialog-ID> element is used to identify this session. If this element is present the <context> element of a <session-policy> container, all properties defined in this container only apply to the identified session. A single document may contain multiple <session-policy> containers, which each contains a different <dialog-ID> element. This way, session-specific policies for different sessions can be contained in one document. If the user agent does not have a session with this dialog-ID, the content of the respective <session-policy> container MUST be ignored.

The <dialog-ID> element is optional in a <context> element. It MUST

contain a <call-ID> and a <local-tag> and MAY contain a <remote-tag> element.

The <call-ID> element contains the call-ID (as defined in [15]) of the session the policies are for.

The <local-tag> element contains the local tag (as defined in [15]) of the session the policies are for.

The <remote-tag> element contains the remote tag (as defined in [15]) of the session the policies are for. If the remote tag element is omitted, the policies apply to all sessions that have the given call-ID and local tag.

Local and remote tags are defined from the viewpoint of the recipient of the document.

4.4.4 The <domain> Element

The <domain> element contains a URI that identifies the domain which has issued this policy.

The <domain> element is optional and MAY occur only once inside a <context> element.

4.4.5 The <contact> Element

The <contact> element contains a contact address (e.g. a SIP URI or email address) under which the issuer of this policy can be reached.

The <contact> element is optional and MAY occur multiple times inside a <context> element.

4.4.6 The <info> Element

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

The <info> element is optional and MAY occur only once inside a <context> element.

4.4.7 The <media-types> Element

The <media-types> element expresses a policy for the use of media types (e.g. audio, video). It defines the media types that must be used, may be used, and must not be used in a session.

This element may have the following attributes (see Section 4.3):

visibility, excluded-policy, direction. The 'excluded-policy' attribute specifies the default policy for all media types that are not listed inside this element.

The <media-types> element is optional in a <session-policy> element and MAY occur multiple times. Multiple <media-types> elements MAY only be present if each element applies to a different set of streams (e.g. one <media-types> for incoming and one for outgoing streams). The <media-types> MUST contain one or more <media-type> elements.

Merging rule: <media-types> containers are merged using the "Multiple Enumerated Value Merging Algorithm" defined in [13].

4.4.8 The <media-type> Element

The <media-type> element defines a policy for the use of the media type identified by this element. The value of this element MUST be the name of a IANA registered media type (see [2]), such as 'audio', 'video', 'text', or 'application'.

This element may have the following attributes (see Section 4.3): policy, q. Media types that have the policy 'mandatory' MUST be used in a session, media types with the policy 'allowed' MAY be used and media types with the policy 'disallowed' MUST NOT be used.

The <media-type> element is mandatory and MAY occur multiple times inside a <media-types> element.

4.4.9 The <codecs> Element

The <codecs> element expresses a policy for the use of codecs. A policy can define that a codec must be used, may be used, or must not be used in a session. A policy MUST allow the use of at least one codec and MUST NOT define more than one mandatory codec for a media type.

This element may have the following attributes (see Section 4.3): visibility, excluded-policy, direction, stream-label. The 'excluded-policy' attribute specifies the default policy for all codecs that

are not listed inside this element.

The <codecs> element is optional in a <session-policy> element and MAY occur multiple times. Multiple <codecs> elements MAY only be present if each element applies to a different set of streams (e.g. one <codecs> for incoming and one for outgoing streams). The <codecs> element MUST contain one or more <codec> elements.

Merging rule: <codecs> containers are merged using the "Multiple Enumerated Value Merging Algorithm" defined in [13].

4.4.10 The <codec> Element

The <codec> element defines a policy for the use of the codec identified by this element. 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".

This element may have the following attributes (see Section 4.3): policy, q. Codecs that have the policy 'mandatory' MUST be used in a session, codecs with the policy 'allowed' MAY be used and codecs with the policy 'disallowed' MUST NOT be used.

The <codec> element is mandatory and MAY occur multiple times inside a <codecs> element.

4.4.11 The <media-intermediary> Element

The <media-intermediary> element expresses a policy for routing a media stream through a media intermediary. The purpose of the <media-intermediary> element is to tell the UA to send the media for a particular stream through an IP address and port on an intermediary. Instead of merely sending the media there, the UA can instead specify a source route, which touches that intermediary, but also any other intermediaries and then the final recipient. Thus, if there are N hops, including the final recipient, there needs to be a way for the media stream to specify N destinations. The way these N destinations should be identified when sending the media stream is expressed using the <int-lroute> element.

This element may have the following attributes (see Section 4.3): visibility, policy, direction, stream-label.

The <media-intermediary> element is optional in a <session-policy> element and MAY occur multiple times. The order of <media-intermediary> element instances is significant. It defines the order in which the media intermediaries must be traversed. The UA sends the media stream to the intermediary listed first, then to the intermediary listed next and so on. The <media-intermediary> element MUST contain one <int-uri> and one <int-lroute> element.

Merging rule: the intermediaries defined in all policies are traversed. In general, local intermediaries should be traversed before remote intermediaries. During the merging process, <media-intermediary> element values from different servers are ordered using the "Closest Value First Merging Algorithm" [13]. The

intermediaries should be traversed in this order.

4.4.12 The <int-uri> Element

The <int-uri> element contains a URI that identifies the IP address and port number of a media intermediary. The UA uses this URI to send its media streams to the intermediary. If a protocol uses multiple subsequent ports (e.g. RTP), the lowest port number SHOULD be included in the URI. All additional port numbers SHOULD be identified in <int-addl-port> elements.

The <int-uri> element occurs exactly once inside a <media-intermediary> element.

4.4.13 The <int-addl-port> Element

If a protocol uses multiple subsequent ports (e.g. RTP), the lowest port number SHOULD be included in the <int-uri> element. All additional port numbers SHOULD be identified in <int-addl-port> elements.

The <int-addl-port> element is optional and MAY occur multiple times inside a <media-intermediary> element.

4.4.14 The <int-lroute> Element

The <int-lroute> element identifies the loose source routing protocol to be used with this intermediary. The value of this element can be one of the following:

- ip-in-ip: IP-in-IP tunneling is used to specify the hops of media traversal. The ultimate destination is specified in the destination IP address of the innermost packet. Each subsequent hop results in another encapsulation, with the destination of that hop in the destination IP address of the packet.
- ip-loose: IP provides a loose routing mechanism that allows the sender of an IP datagram to specify a set of IP addresses that are to be visited on the way before reaching the final destination.
- turn: TURN provides a mechanism for inserting a media relay into the path. Although the main purpose of TURN is NAT traversal, it is possible for a TURN relay to perform other media intermediary functionalities. The user agent establishes a binding on the TURN server and uses this binding to transmit and receive media.
- media-specific: media protocols can provide their own loose routing mechanism. If that is the case, the loose routing mechanism of that protocol is used. As an example, SIP provides its own loose routing mechanisms with the Route header. It can be used to direct an instant message using the SIP MESSAGE method

through a set of intermediaries.

- none: if there is no loose-routing mechanism available, the media is just sent to the first media intermediary listed in the header. Note that this requires the intermediary to know where to forward the packets to. Such a route must be set up in the intermediary through other means. For example, with session-specific policies, the policy server can extract the destination address from the session description.

The <int-lroute> element occurs exactly once inside a <media-intermediary> element.

4.4.15 The <max-bandwidth> Element

The <max-bandwidth> element contains the maximum bandwidth in kilobits per second an entity can use for its media streams.

This element may have the following attributes (see Section 4.3):

visibility, policy, direction, media-type.

The <max-bandwidth> element is optional and MAY occur multiple times inside a <session-policy> element. If it occurs multiple times, each instance MUST apply to different media streams (i.e. one <max-bandwidth> element for outgoing and one for incoming streams).

Merging rule: the lowest max-bandwidth value is used.

4.4.16 The <qos-dscp> Element

The <qos-dscp> element contains an Differentiated Services Codepoint (DSCP) [10] that should be used to populate the IP DS field of media packets. The <qos-dscp> contains an integer value that represents a 6 bit field and therefore ranges from 0 to 63.

This element may have the following attributes (see Section 4.3): visibility, policy, direction, stream-label, media-type.

The <qos-dscp> element is optional and MAY occur multiple times inside a <session-policy> element. If it occurs multiple times, each instance MUST apply to a different media stream (i.e. one <qos-dscp> element for audio and one for video streams).

Merging rule: the domain that is first traversed by the media stream has precedence and its DSCP value is used. During the merging process, <qos-dscp> element values from different servers are ordered using the "Closest Value First Merging Algorithm" [13]. The DSCP value from the closest server is used.

4.4.17 Other Elements

A number of additional elements have been proposed for a policy language. These elements are deemed to be outside the scope of a basic media policy format. However, they may be defined in extensions of BMPF or other profile data sets.

- maximum number of streams
- maximum number of sessions
- maximum number of streams per session
- maximum bandwidth per session

- maximum bandwidth per stream
- external address and port
- media transport protocol
- outbound proxy
- SIP methods
- SIP option tags
- SIP transport protocol
- body disposition
- body format
- body encryption

4.5 Example

The following example describes a policy that requires the use of audio, allows the use of video and prohibits the use of other media types. It allows the use of any codec except G.723 and G.729. The policy also inserts a media intermediary into outgoing media streams.


```
<property-set>
  <session-policy>
    <context>
      <domain>example.com</domain>
```

```
        <contact>sip:policy_manager@example.com</contact>
        <info>Access network policies</info>
      </context>
      <media-types excluded-policy="disallow">
        <media-type policy="mandatory">audio</media-type>
        <media-type policy="allow">video</media-type>
      </media-types>
      <codecs excluded-policy="allow">
        <codec policy="disallow">G729</codec>
        <codec policy="disallow">G723</codec>
      </codecs>
      <media-intermediary direction="sendonly" policy="mandatory">
        <int-uri>192.0.2.0:6000</int-uri>
        <int-addl-port>6001</int-addl-port>
        <int-lroute>ip-in-ip</int-lroute>
      </media-intermediary>
    </session-policy>
  </property-set>
```

4.6 Schema Definition

```
  <?xml version="1.0" encoding="UTF-8"?>
  <xs:schema targetNamespace="urn:ietf:params:xml:ns:mediadataset"
        xmlns:tns="urn:ietf:params:xml:ns:mediadataset"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:spds="http://sipfoundry.org/schema/profile-data-sets-00">


    <xs:attributeGroup name="single_stream_attributes" >
      <xs:attribute name="stream-label"
                    type="xs:string" use="optional"/>
    </xs:attributeGroup>

    <xs:attributeGroup name="media_type_attributes" >
      <xs:attribute name="media-type"
                    type="xs:string" use="optional"/>
    </xs:attributeGroup>


    <xs:element name="session-policy">
      <xs:complexType>
        <xs:sequence>
```

```
        <xs:element ref="tns:context"
          minOccurs="0" maxOccurs="1"/>
        <xs:element ref="tns:media-types"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="tns:codecs"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="tns:media-intermediary"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="tns:max-bandwidth"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="tns:qos-dscp"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>


  <xs:element name="context">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:dialog-ID"
          minOccurs="0" maxOccurs="1"/>
        <xs:element name="domain" type="xs:anyURI" minOccurs="0"
          maxOccurs="1"/>
        <xs:element name="contact" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:element name="info" type="xs:string"
           minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="media-types"
              substitutionGroup="spds:setting_container">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:media-type"
          minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attributeGroup ref="spds:directional_setting_attributes" />
    </xs:complexType>
  </xs:element>

  <xs:element name="codecs"
              substitutionGroup="spds:setting_container">
    <xs:complexType>
      <xs:sequence>
```

```
          <xs:element ref="tns:codec"
```



```
            minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attributeGroup ref="spds:directional_setting_attributes" />
        <xs:attributeGroup ref="tns:single_stream_attributes" />
      </xs:complexType>
    </xs:element>

    <xs:element name="media-intermediary"
                substitutionGroup="spds:setting">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="int-uri" type="xs:anyURI"
            minOccurs="1" maxOccurs="1"/>
          <xs:element name="int-addl-port"
            type="xs:positiveInteger"
            minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="int-lroute" type="tns:int-lroute"
            minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attributeGroup ref="spds:directional_setting_attributes" />
        <xs:attributeGroup ref="tns:single_stream_attributes" />
      </xs:complexType>
    </xs:element>

    <xs:element name="max-bandwidth"
                substitutionGroup="spds:setting">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:positiveInteger" />
        </xs:simpleContent>
        <xs:attributeGroup ref="spds:directional_setting_attributes" />
        <xs:attributeGroup ref="tns:media_type_attributes" />
      </xs:complexType>
    </xs:element>

    <xs:element name="qos-dscp"
                substitutionGroup="spds:setting">
      <xs:complexType>
        <xs:simpleContent>
          <xs:restriction base="xs:integer" >
```

```
            <xs:minInclusive value="0" />
            <xs:maxInclusive value="63" />
          </xs:restriction>
        </xs:simpleContent>
        <xs:attributeGroup ref="spds:directional_setting_attributes" />
        <xs:attributeGroup ref="tns:single_stream_attributes" />
        <xs:attributeGroup ref="tns:media_type_attributes" />
      </xs:complexType>
```



```
    </xs:element>


    <xs:element name="dialog-ID">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="call-ID" type="xs:string"
            minOccurs="1" maxOccurs="1"/>
          <xs:element name="local-tag" type="xs:string"
            minOccurs="1" maxOccurs="1"/>
          <xs:element name="remote-tag" type="xs:string"
            minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:element name="media-type"
                substitutionGroup="spds:setting">
      <xs:complexType>
        <xs:simpleContent>
          <xs:restriction base="xs:string" />
        </xs:simpleContent>
        <xs:attributeGroup ref="spds:multi_setting_attributes" />
      </xs:complexType>
    </xs:element>

    <xs:element name="codec"
                substitutionGroup="spds:setting">
      <xs:complexType>
        <xs:simpleContent>
          <xs:restriction base="xs:string" />
        </xs:simpleContent>
        <xs:attributeGroup ref="spds:multi_setting_attributes" />
```

```
      </xs:complexType>
    </xs:element>

    <xs:simpleType name="int-lroute">
      <xs:restriction base="xs:string">
        <xs:enumeration value="ip-in-ip"/>
        <xs:enumeration value="ip-loose"/>
        <xs:enumeration value="turn"/>
        <xs:enumeration value="media-specific"/>
        <xs:enumeration value="none"/>
      </xs:restriction>
    </xs:simpleType>

  </xs:schema>
```


5. 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.

6. IANA Considerations

This document registers a new MIME type, application/ session-policy+xml, and registers a new XML namespace.

6.1 MIME Registration for application/session-policy+xml

MIME media type name: application

MIME subtype name: session-policy+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [9].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [9].

Security considerations: See Section 10 of RFC 3023 [9] and Section 5 of this specification.

Interoperability considerations: none.

Published specification: This document.

Applications which use this media type: This document type has been used to download the session policy of a domain to SIP user agents.

Additional Information:

Magic Number: None

File Extension: .wif or .xml

Macintosh file type code: "TEXT"

Personal and email address for further information: Volker Hilt, <volkerh@bell-labs.com>

Intended usage: COMMON

Author/Change controller: The IETF.

6.2 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:mediadataset

This section registers a new XML namespace, as per the guidelines in [5]

URI: The URI for this namespace is urn:ietf:params:xml:ns:mediadataset.

Registrant Contact: IETF, SIPPING working group, <sipping@ietf.org>, Volker Hilt, <volkerh@bell-labs.com>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
          "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1"/>
  <title>Session Policy Namespace</title>
</head>
<body>
  <h1>Namespace for Session Policy Information</h1>
  <h2>urn:ietf:params:xml:ns:mediadataset</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

7. References

7.1 Normative References

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

[2] Handley, M., "SDP: Session Description Protocol", draft-ietf-mmusic-sdp-new-24 (work in progress), February 2005.

[3] Hilt, V., Rosenberg, J., and G. Camarillo, "Media Type Extension Negotiation in the Session Initiation Protocol (SIP) Accept Header Field", draft-hilt-sip-ext-neg-00 (work in progress), January 2005.

[4] Hollander, D., Bray, T., and A. Layman, "Namespaces in XML", W3C REC REC-xml-names-19990114, January 1999.

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

[6] Levin, O. and G. Camarillo, "The SDP (Session Description Protocol) Label Attribute", draft-ietf-mmusic-sdp-media-label-01 (work in progress), January 2005.

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

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

[9] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[10] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[11] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[12] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-06 (work in progress), February 2005.

[13] Petrie, D., Lawrence, S., Dolly, M., and V. Hilt, "A Schema and Guidelines for Defining Session Initiation Protocol User Agent Profile Data Sets", draft-petrie-sipping-profile-datasets-02 (work in progress), April 2005.

[14] Rosenberg, J., "Traversal Using Relay NAT (TURN)", draft-rosenberg-midcom-turn-07 (work in progress), February 2005.

[15] 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.

[16] Yergeau, F., Paoli, J., Sperberg-McQueen, C., Bray, T., and E. Maler, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C REC REC-xml-20040204, February 2004.

7.2 Informative References

[17] Hilt, V., Camarillo, G., and J. Rosenberg, "A Framework for Session-Specific Session Policies in the Session Initiation Protocol (SIP)", draft-hilt-sipping-session-spec-policy-01 (work in progress), October 2004.

[18] Rosenberg, J., "Requirements for Session Policy for the Session Initiation Protocol (SIP)", draft-ietf-sipping-session-policy-req-02 (work in progress), July 2004.

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@cisco.com

Appendix A. Acknowledgements

Many thanks to Allison Mankin, Dan Petrie and Martin Dolly for the

great discussions and suggestions. A big thanks also to everyone who contributed by providing feedback on the mailing list and in IETF meetings.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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