SIPPING Working Group
Internet-Draft
Expires: August 10, 2005

V. Hilt
Bell Labs/Lucent Technologies
G. Camarillo
Ericsson
J. Rosenberg
Cisco Systems
February 9, 2005

Session Initiation Protocol (SIP) Session Policies - Document Format and Session-Independent Delivery Mechanism
draft-ietf-sipping-session-indep-policy-02

Status of this Memo

Copyright Notice

Abstract

This draft defines a delivery mechanism for SIP session policies that is independent of a specific SIP session. It also defines the Basic Session Policy Format (BSPF), which is a minimal, XML-based format

for session policies.

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 want to restrict codec usage on the network to lower rate codecs or disallow the use of high bandwidth media types such as video.

In another example, a network has a resource reservation infrastructure in place, which enables user agents to request Quality of Service (QoS) for media streams. With session policies, the network can tell user agents that a QoS infrastructure is present and ask user agents to use specific parameters or provide certain credentials when requesting QoS.

In a third example, a user has subscribed to a service that requires the media streams to be routed through a media intermediary. The service provider would like to tell the user agent to direct the media streams to this intermediary and to use a certain source routing scheme (e.g. IP-in-IP tunneling). Knowing this policy enables the user to use this service in any network from which the intermediary can be reached.

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

[3] 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. An example is a policy which prohibits the use of high-bandwidth codecs. 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 the Basic Session Policy Format (BSPF). BSPF is a minimal session policy format aimed at achieving interoperability between different user agents and policy servers. This format introduces a common data model and defines a basic set of policy elements. The format is based on XML [16] and can be extended using XML extension mechanisms. The document format is independent of the policy delivery mechanism and can be used for 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 policy servers. The following sections describe the subscription to the session-independent policies relevant for a UA.

3.1 Subscriber Behavior

A UA can express interest in session-independent policies by subscribing to session policies using the mechanism defined in [12]. If the UA has already received the URIs of all relevant session

policy servers (e.g., through configuration) it SHOULD 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. 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 "local" (for subscriptions to the network domain) profile-types [12]. A UA SHOULD modify these subscriptions as described below in following events:

- The UA changes the registration status of one of its AoR. This occurs, for example, when a UA starts up and registers its AoRs, when it shuts down and deregisters AoRs, or when a new AoR is added to a UA. In these events, the UA SHOULD establish subscriptions for each new AoR using the "user" and the "local" profile-types. It SHOULD terminate the subscriptions for all AoRs that have been removed.
- The domain the UA is connected to changes. The UA SHOULD create a new subscription for each AoR using the "local" profile-type. It SHOULD terminate all existing subscriptions for the "local" profile-type. It does not need to change the subscriptions for "user" profiles.

If a subscriber is unable to successfully 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 MUST 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 [4].

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. If the notifier receives a SUBSCRIBE request, which does not list all MIME types and MIME type extensions in the Accept header that are needed for policies, it MUST reject the request with a 406 response. A policy format is needed, if the network has policies in this format that must be used by the UA. The notifier SHOULD NOT return a 406 if an unsupported format contains optional policies.

4. Policy Format Design

The following sections describe design considerations for an XML-based model for session policies.

4.1 Policy Model

Session policies influence aspects of a SIP session by defining constraints. A constraint impacts a specific aspect of a SIP session (e.g. the codecs that can be used in this session). Policy constraints are modeled as XML elements. Each policy element expresses a certain constraint. Policy elements can contain a simple value or act as a container, which holds multiple alternative values for this policy.

Elements that express policies have a 'policy' attribute. This attribute defines the constraining properties of the XML element. The following values are defined for the 'policy' attribute:

- mandatory: the value contained in the element is mandatory and MUST be used in sessions. This is the default value that is used if the 'policy' attribute is omitted.
- allow: the value contained in the element is allowed and MAY be used in sessions.
- disallow: the value contained in the element is forbidden and MUST NOT be used in sessions.

Polices consisting of one single value can be expressed by a simple policy element. The following is an example of a policy defining an upper limit for media bandwidth:

```
<max-bandwidth>80</max-bandwidth>
```

Policies consisting of multiple values can be expressed using a

container element. The container contains multiple elements, which define possible values. The policy attribute of the container specifies the policy that applies to all values not listed in the container. The policy attribute of each element in the container defines the policy for that item. The following example shows a policy that requires the media type audio and allows video in sessions:

```
<media-types policy="disallow">
  <media-type policy="mandatory">audio</media-type>
  <media-type policy="allow">video</media-type>
</media-types>
```

4.2 Unidirectional Policies

Some policies only affect media streams flowing into one direction, e.g., only outgoing streams. Unidirectional policies can be expressed by adding a 'direction' attribute to the respective policy element.

The 'direction' attribute can have the following values:

- recvonly: the policy only applies to incoming streams.
- sendonly: the policy only applies to outgoing streams.
- sendrecv: the policy applies to streams in both directions. This is the default value that is used if the 'direction' attribute is omitted.

4.3 Per-Stream Policies

Policies can be specific to a certain media stream. The stream to which a policy applies to must be identifiable through a label [7]. Per-stream policies can be expressed by adding a 'label' attribute to the respective policy element. Such a policy only applies to the identified stream. If the label value is unknown to the recipient, the policy must be ignored.

Per-stream policies require that the policy server has access to the session description in order to extract the stream label. For this reason, per-stream policies are typically used in session-specific policies.

4.4 Merging Policies

A UA may receive policy documents from multiple sources, which need to be merged into a single combined policy document the UA can work

with.

Policies that define a single value (e.g. maximum bandwidth) require the selection of one value during the merging process. The selection criteria must be defined individually for each element (e.g. select lowest value) in the schema definition.

Policies that allow multiple values can be merged by combining all values and adjusting the 'policy' attribute for values that exist in both documents. Table 1 provides a matrix for merging the 'policy' attributes. Additional merging rules may be required for some elements. They must be specified in the schema definition.

Some constellations are not feasible and constitute a policy conflict that can not be resolved automatically. If the conflicting policies are enforced by the network, the UA may experience difficulties when setting up a session.

pol 1 \ pol 2	mandatory	allow	disallow
mandatory	mandatory	mandatory	conflict!
allow	mandatory	allow	disallow
disallow	conflict!	disallow	disallow

Table 1: merging policies.

The combined policy MUST again be valid and well-formed according to policy schema definitions. A policy conflict occurs if the combined policy is not a well-formed document after the merging process is completed.

5. Basic Session Policy Format

The Basic Session Policy Format (BSPF) defines the structure of and a root element for session policy documents. It provides a minimal set of policy elements as required by [14]. To enable interoperability between UAs and policy servers, this format MUST be supported by all UAs compliant to this specification.

> Note: It is the goal to keep this specification aligned with the schema for user agent profile data sets [13] to simplify the processing of policy and configuration data.

5.1 MIME Type and Namespace

The MIME type for the Basic Session Policy Format is:

application/session-policy+xml

This specification makes use of XML namespaces [5]. The namespace URIs for schemas defined in this specification are URNs [8], using the namespace identifier 'ietf' defined by [9] and extended by [6]. The namespace URN for the BSPF schema is:

urn:ietf:params:xml:ns:sessionpolicy

5.2 Extensibility

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

5.3 XML Format Definition

A BSPF 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. BSPF documents MUST be based on XML 1.0 and MUST be encoded using UTF-8.

5.3.1 The <session-policy> Element

The root element of a BSPF document is the <session-policy> element.

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

5.3.2 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 <domain>, multiple <contact>, a <info>, and multiple <entity> elements.

Merging policies: the <context> element is not subject to merging. Information in the context element may be used to assist the user if a policy conflict occurs. Policies that affect different entities (e.g. different AoRs) on a user agent and therefore have different <entity> values do not need to be merged. A policy affecting all entities on a user agent (i.e. no <entity> element is specified) must be merged with the policy for a specific entity.

5.3.3 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.

5.3.4 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.

5.3.5 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.

5.3.6 The <entity> Element

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

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

5.3.7 The <media-types> Element

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

This element has an mandatory 'policy' attribute as defined in Section 4.1. The 'policy' attribute specifies the default policy for all media types that are not listed inside this element.

This element has an optional 'direction' attribute as defined in Section 4.2.

The <media-types> element is optional in a <session-policy> element and MAY occur multiple times. It MUST contain one or more <media-type> elements.

Merging policies: the 'policy' attribute of the <media-types> element and <media-type> elements with the same value is adjusted according to Table 1.

5.3.8 The <media-type> Element

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

This element has a mandatory 'policy' attribute as defined in Section 4.1.

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

5.3.9 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 has a mandatory 'policy' attribute as defined in Section 4.1. The 'policy' attribute specifies the default policy for all codecs that are not listed inside this element.

This element has an optional 'direction' attribute as defined in Section 4.2.

This element has an optional 'label' attribute as defined in Section 4.3.

The <codecs> element is optional in a <session-policy> element and MAY occur multiple times. It MUST contain one or more <codec> elements.

Merging policies: the 'policy' attribute of the <codecs> element and <codec> elements with the same value is adjusted according to Table 1.

5.3.10 The <codec> Element

The <codec> element defines a policy for the use of the codec identified by this elements value. This value MUST be the name of a registered MIME type for a encoding (see [2]), such as "PCMA", "G729", or "H263".

This element has a mandatory 'policy' attribute as defined in Section 4.1.

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

5.3.11 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 of 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 has a mandatory 'policy' attribute as defined in Section 4.1). This attribute defines whether routing media streams through this intermediary is mandatory or allowed.

This element has an optional 'label' attribute as defined in Section 4.3.

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 policies: the intermediaries defined in all policies are traversed. For session-independent policies, intermediaries received through a subscription using the "local" profile-type are traversed before those received through a "user" profile-type subscription. For session-specific policies, intermediaries are traversed in the order in which policy URIs are received (i.e.

local intermediaries first).

5.3.12 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) only the lowest port number needs to be identified.

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

5.3.13 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 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.
- 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.

5.3.14 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 has an optional 'direction' attribute as defined in Section 4.2. If the direction attribute is present, the <max-bandwidth> element contains the bandwidth available in the identified direction.

The <max-bandwidth> element is optional in a <session-policy> element and MAY occur only once.

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

Open issue: The maximum bandwidth policy is not part of the policy requirements. Should it be part of BSPF?

5.3.15 The <qos> Element

Open issue: what needs to go in here?

5.3.16 Open Issue: Other Elements

A number of additional elements have been proposed for a policy language:

- 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

Is it desirable to add any of these to the BSPF format? Some of these items could become part of an extension to BSPF.

5.4 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.

```
   <session-policy>
     <context>
       <domain>example.com</domain>
       <contact>sip:policy_manager@example.com</contact>
       <info>Access network policies</info>
     </context>
     <media-types policy="disallow">
       <media-type policy="mandatory">audio</media-type>
       <media-type policy="allow">video</media-type>
     </media-types>
     <codecs policy="allow">
       <codec policy="disallow">G729</codec>
       <codec policy="disallow">G723</codec>
     </codecs>
     <media-intermediary direction="sendonly" policy="mandatory">
       <int-uri>123.234.123.234:6000</int-uri>
       <int-lroute>ip-in-ip</int-lroute>
     </media-intermediary>
   </session-policy>
```

5.5 Schema Definition

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

     <xs:element name="session-policy" type="tns:session-policy"/>

     <xs:complexType name="session-policy">
       <xs:sequence>
         <xs:element name="context" type="tns:context" minOccurs="0"
           maxOccurs="1"/>
         <xs:element name="media-types" type="tns:media-types"
           minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="codecs" type="tns:codecs" minOccurs="0"
           maxOccurs="unbounded"/>
         <xs:element name="media-intermediary"
           type="tns:media-intermediary" minOccurs="0"
           maxOccurs="unbounded"/>
         <xs:element name="max-bandwidth" type="tns:max-bandwidth"
           minOccurs="0" maxOccurs="unbounded"/>
       </xs:sequence>
```

```
    </xs:complexType>

    <xs:complexType name="context">
      <xs:sequence>
        <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:element name="entity" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="media-types">
      <xs:sequence>
        <xs:element name="media-type" type="tns:media-type" minOccurs="1"
          maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="policy" type="tns:policyValue"
        use="required"/>
      <xs:attribute name="direction" type="tns:directionValue"
        use="optional" default="sendrecv"/>
    </xs:complexType>

    <xs:complexType name="codecs">
      <xs:sequence>
        <xs:element name="codec" type="tns:codec" minOccurs="1"
          maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="policy" type="tns:policyValue"
        use="required"/>
      <xs:attribute name="direction" type="tns:directionValue"
        use="optional" default="sendrecv"/>
      <xs:attribute name="label" type="xs:string" use="optional"/>
    </xs:complexType>

    <xs:complexType name="media-intermediary">
      <xs:sequence>
        <xs:element name="int-uri" type="xs:anyURI" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="int-lroute" type="tns:int-lroute" minOccurs="1"
          maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="policy" type="tns:policyValue"
        use="required"/>
```

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

    <xs:complexType name="max-bandwidth">
      <xs:simpleContent>
        <xs:extension base="xs:positiveInteger">
          <xs:attribute name="policy" type="tns:policyValue"
            use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="media-type">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="policy" type="tns:policyValue"
            use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="codec">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="policy" type="tns:policyValue"
            use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

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

    <xs:simpleType name="policyValue">
      <xs:restriction base="xs:string">
        <xs:enumeration value="mandatory"/>
        <xs:enumeration value="allow"/>
        <xs:enumeration value="disallow"/>
      </xs:restriction>
    </xs:simpleType>
```

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

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

7.1 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 [10].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [10].

Security considerations: See Section 10 of RFC 3023 [10] and Section 6 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.

7.2 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:sessionpolicy

This section registers a new XML namespace, as per the guidelines in [6]

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

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:sessionpolicy</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

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)", draft-hilt-sipping-session-spec-policy-01 (work in progress), October 2004.

[4] 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.

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

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

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

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

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

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

[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-05 (work in progress), October 2004.

[13] Petrie, D., "A Schema for Session Initiation Protocol User Agent Profile Data Sets",

draft-petrie-sipping-profile-datasets-00 (work in progress), July 2004.

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

[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.

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 for the discussions and the suggestions

for this draft. Many thanks also to Dan Petrie and Martin Dolly.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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