SIPPING Working Group
Internet-Draft
Expires: October 6, 2006

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

April 4, 2006

A User Agent Profile Data Set for Media Policy
draft-ietf-sipping-media-policy-dataset-01

Status of this Memo

Copyright Notice

Abstract

This specification defines a document format for media properties of Session Initiation Protocol (SIP) sessions, such as the codecs or media types to be used. This format can used to define media properties in SIP user agent profile data sets and SIP session policies. It extends the Schema for SIP User Agent Profile Data

Sets.

Table of Contents

1. Introduction

The Framework for Session Initiation Protocol (SIP) [18] User Agent Profile Delivery [16] and the Framework for SIP Session Policies [15] define mechanisms to convey session policies and configuration information from a network server to a user agent. An important part of this information are properties that define media-related aspects of SIP sessions. These properties include, for example, codecs and media-types to be used, media-intermediaries to be traversed or the maximum bandwidth available in a session.

This draft defines a document format for media properties of SIP sessions, the Media Policy Dataset Format (MPDF). This format can be used to define configuration data sets and session policies. The MPDF format is based on XML [14] and extends the Schema for SIP User Agent Profile Data Sets [12] by specifying a data set for media properties. The format also satisfies the requirements of a minimal set of media-level session policy elements as described in [17]. It can be extended through the XML extension mechanisms if additional media properties are needed.

A MPDF document MUST be well-formed and MUST be valid according to schemas, including extension schemas, available to the validator and applicable to the XML document. MPDF documents MUST be based on XML 1.0 and MUST be encoded using UTF-8.

A user agent may receive multiple MPDF documents from different sources. These documents need to merged into a single document the user agent can work with. General rules for merging MPDF documents are described in [12]. Specific merging rules for each of the MPDF elements are described below.

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. 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 MPDF schema is:

urn:ietf:params:xml:ns:mediadataset

The MIME type for the Media Policy Dataset Format is:

application/session-policy+xml

OPEN ISSUE: do we need a separate MIME type might or is the MIME type of the Schema for SIP User Agent Profile Data Sets sufficient.

4. Extensibility

The MPDF format is an extension of the Schema for SIP User Agent Profile Data Sets [12]. Elements from the MPDF namespace can be used in conjunction with elements from other extensions of this schema.

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

5. 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 [12]:

- 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.

5.1. 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.

5.2. 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 [3]), such as 'audio', 'video', 'text', or 'application'.

6. Elements

The following elements are defined for the MPDF format.

6.1. The <session-policy> Element

The <session-policy> element is a container for media policy elements. It MAY occur multiple times inside a <property_set> [12] element.

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

> OPEN ISSUE: the <session-policy> is a container for policies. This may not be needed if policy element are directly placed in a <property_set> element. In this case, the <context> element needs to be a child of the <property_set> element.

6.2. The <context> Element

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

The <context> element is optional in a <session-policy> element. It

MAY contain a <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.

6.2.1. 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.

6.2.2. 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.

6.2.3. 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.

6.3. 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 5): 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 [12].

6.3.1. 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 [3]), such as 'audio', 'video', 'text', or 'application'.

This element may have the following attributes (see Section 5): 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.

6.4. 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 5): 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 [12].

6.4.1. The <codec> Element

The <codec> element defines a policy for the use of the codec identified by this element. A codec is identified based on a registered MIME type [2] using media-type and subtype (e.g. audio/PCMA or video/H263) and possibly additional registered MIME type parameters.

This element may have the following attributes (see Section 5): 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. The <codec> element MUST contain one <mime-type> element and MAY contain multiple optional <mime-parameter> elements.

6.4.1.1. The <mime-type> Element

The <mime-type> element identifies a codec. The value of this element MUST be a combination of a registered MIME media-type and subtype [2] separated by a "/" (e.g. audio/PCMA, audio/G726-16, video/H263).

The <mime-type> element occurs exactly once inside a <codec> element.

6.4.1.2. The <mime-parameter> Element

The <mime-parameter> element may be needed for some codecs to identify a particular encoding or profile. The value of this element MUST be a name-value pair containing the name and the value of a registered MIME type parameter for the codec [2]. The name and value are separated by a "=". For example, the parameter "profile=0" can be used to specify a specific profile for the codec "video/H263-2000".

The <mime-parameter> element is optional and MAY occur multiple times inside a <codec> element.

6.5. The <media-intermediaries> Element

The <media-intermediaries> element expresses a policy for routing a media stream through a media intermediary. The purpose of the <media-intermediaries> element is to tell the UA to send a media stream through one (or a chain of) media intermediaries. Instead of sending the media directly to its final destination, the UA instead specifies a source route, which touches each intermediary and then reaches the final recipient. If there are N hops, including the final recipient, there needs to be a way for the media stream to specify N destinations.

The <media-intermediaries> element is a container that lists all media intermediaries to be traversed. Media intermediaries should be traversed in the order in which they appear in this list. The topmost entry should be traversed first, the last entry should be traversed last.

Different types of intermediaries exist. These intermediaries are not necessarily interoperable and it may not be possible to chain them in an arbitrary order. A <media-intermediaries> element SHOULD therefore only contain intermediary elements of the same type.

This element may have the following attributes (see Section 5): visibility, policy, direction, stream-label.

The <media-intermediaries> element is optional in a <session-policy> element and MAY occur multiple times. Multiple <media-intermediaries> elements MAY only be present if each element applies to a different set of streams (e.g. one <media-intermediaries> element for incoming and one for outgoing streams). The <media-intermediaries> element MUST contain one or more of the following elements (all of the same type): <configured-intermediary>, <turn-intermediary>, <ipinip-intermediary>, and <iploose-intermediary>.

> 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-intermediaries> element values from different servers are ordered using the "Closest Value First Merging Algorithm" [12]. The intermediaries should be traversed in this order.
>
> Note: it is not intended that the <media-intermediaries> element replaces connectivity discovery mechanisms such as ICE. Instead of finding media relays that provide connectivity, this element defines a policy for media intermediaries that should be traversed. The set of intermediaries defined in the <media-intermediaries> element and the ones discovered through ICE may overlap but don't have to.

6.5.1. The <configured-intermediary> Element

A configured intermediary relies on configured forwarding rules. The user agent simply sends media to the first media intermediary listed. It can assume that this media intermediary has been configured with a forwarding rule for the media stream and knows where to forward the packets to. The configuration of forwarding rules in the intermediary must be done through other means.

The <configured-intermediary> element is optional and MAY occur multiple times inside a <media-intermediaries> element. The <configured-intermediary> element MUST contain one <int-uri> element and MAY contain multiple optional <int-addl-port> elements.

6.5.1.1. 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 the following elements: <configured-intermediary>, <turn-intermediary>, <ipinip-intermediary>, and <iploose-intermediary>.

6.5.1.2. 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 the following elements: <configured-intermediary>, <turn-intermediary>, <ipinip-intermediary>, and <iploose-intermediary>.

6.5.2. The <turn-intermediary> Element

The TURN [13] protocol provides a mechanism for inserting a relay into the media 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.

The <turn-intermediary> element is optional and MAY occur multiple times inside a <media-intermediaries> element. The <turn-intermediary> element MUST contain one <int-uri> element and MAY contain multiple optional <int-addl-port> elements and one optional <shared-secret> element.

6.5.2.1. The <shared-secret> Element

The <shared-secret> element contains the shared secret needed to authenticate at the TURN server.

The <shared-secret> element is optional and MAY occur only once inside the <turn-intermediary> element.

6.5.3. The <ipinip-intermediary> Element

For these intermediaries, IP-in-IP tunneling [11] is used to specify the hops of media intermediary 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.

The <ipinip-intermediary> element is optional and MAY occur multiple times inside a <media-intermediaries> element. The <ipinip-intermediary> element MUST contain one <int-uri> element and MAY contain multiple optional <int-addl-port> elements.

6.5.4. The <iploose-intermediary> Element

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.

The <iploose-intermediary> element is optional and MAY occur multiple times inside a <media-intermediaries> element. The <iploose-intermediary> element MUST contain one <int-uri> element and MAY contain multiple optional <int-addl-port> elements.

6.6. 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 5): 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.

6.7. The <qos-dscp> Element

The <qos-dscp> element contains an Differentiated Services Codepoint (DSCP) [10] value 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 5): 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" [12]. The DSCP value from the closest server is used.

6.8. The <local-ports> Element

Domains often require that a user agent only uses ports in a certain range for media streams. The <local-ports> element defines a policy for the ports a user agent can use for media. The value of this element consists of a start port and an end port separated by a "-". The start/end port is the first/last port that can be used.

This element may have the following attributes (see Section 5): visibility.

The <local-ports> element is optional and MAY occur multiple times inside a <session-policy> element.

Merging rule: the domain that is first traversed by the media stream has precedence and its local ports value is used. During the merging process, <local-ports> element values from different servers are ordered using the "Closest Value First Merging Algorithm" [12]. The value from the closest server is used.

6.9. Other Elements

A number of additional elements have been proposed for a policy language. These elements are deemed to be outside the scope of this media policy format. However, they may be defined in extensions of MPDF 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

7. 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 using IP-in-IP tunneling.

```
<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">
        <mime-type>audio/G729</mime-type>
      </codec>
      <codec policy="disallow">
        <mime-type>audio/G723</mime-type>
      </codec>
    </codecs>
    <media-intermediaries direction="sendonly" policy="mandatory">
      <ipinip-intermediary>
        <int-uri>192.0.2.0:6000</int-uri>
        <int-addl-port>6001</int-addl-port>
      </ipinip-intermediary>
    </media-intermediaries>
  </session-policy>
</property-set>
```

8. 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-intermediaries"
          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:element ref="tns:local-ports"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>


  <xs:element name="context">
    <xs:complexType>
      <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: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-intermediaries"
            substitutionGroup="spds:setting">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tns:configured-intermediary"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="tns:turn-intermediary"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="tns:ipinip-intermediary"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="tns:iploose-intermediary"
        minOccurs="0" 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="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="local-ports"
                substitutionGroup="spds:setting">
      <xs:complexType>
        <xs:simpleContent>
          <xs:restriction base="xs:string" />
        </xs:simpleContent>
      </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:sequence>
          <xs:element name="mime-type" type="xs:string"
            minOccurs="1" maxOccurs="1"/>
          <xs:element name="mime-parameter" type="xs:string"
            minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attributeGroup ref="spds:multi_setting_attributes" />
      </xs:complexType>
    </xs:element>

    <xs:element name="configured-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:sequence>
        <xs:attributeGroup ref="spds:multi_setting_attributes" />
      </xs:complexType>
    </xs:element>

    <xs:element name="turn-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="shared-secret" type="xs:string"
            minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attributeGroup ref="spds:multi_setting_attributes" />
      </xs:complexType>
    </xs:element>

    <xs:element name="ipinip-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:sequence>
        <xs:attributeGroup ref="spds:multi_setting_attributes" />
      </xs:complexType>
    </xs:element>

    <xs:element name="iploose-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:sequence>
        <xs:attributeGroup ref="spds:multi_setting_attributes" />
      </xs:complexType>
    </xs:element>


  </xs:schema>
```

9. 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.

10. IANA Considerations

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

10.1. 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 9 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.

10.2. 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
```

Appendix A. Acknowledgements

Many thanks to Allison Mankin, Dan Petrie and Martin Dolly for the great discussions and suggestions. Many thanks also to everyone who contributed by providing feedback.

11. References

11.1. Normative References

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

[2] Casner, S. and P. Hoschka, "MIME Type Registration of RTP Payload Formats", RFC 3555, July 2003.

[3] Handley, M., "SDP: Session Description Protocol", draft-ietf-mmusic-sdp-new-26 (work in progress), January 2006.

[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., 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-03 (work in progress), October 2005.

[13] Rosenberg, J., "Traversal Using Relay NAT (TURN)", draft-rosenberg-midcom-turn-08 (work in progress), September 2005.

[14] 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.

11.2. Informative References

[15] Hilt, V., Camarillo, G., and J. Rosenberg, "A Framework for Session Initiation Protocol (SIP) Session Policies", draft-ietf-sipping-session-policy-framework-00 (work in progress), March 2006.

[16] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-08 (work in progress), March 2006.

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

[18] 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@cisco.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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