User Agent Capability Extension to Presence Information Data Format (PIDF)
draft-ietf-simple-prescaps-ext-05

Abstract

Interoperation of instant messaging and presence systems has been defined in the IMPP working group. The IMPP WG has come up with baseline interoperable operations and formats for presence and instant messaging systems. This memo defines an extension to represent RFC3840 capabilities in the Presence Information Document Format (PIDF) compliant presence documents.
Table of Contents

1. Introduction ................................................. 4
   1.1. Motivation ............................................ 4
   1.2. Scope ................................................ 5
2. Conventions ................................................... 5
3. Extension for "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)" in PIDF documents .......... 5
   3.1. Overview of operation ................................ 5
   3.2. Service capability element ............................. 6
       3.2.1. <servcaps> element ................................ 6
       3.2.2. <audio> element ..................................... 6
       3.2.3. <application> element ............................... 6
       3.2.4. <data> element ...................................... 7
       3.2.5. <control> element ................................... 7
       3.2.6. <video> element .................................... 7
       3.2.7. <text> element ...................................... 7
       3.2.8. <message> element ................................... 8
       3.2.9. <type> element ...................................... 8
       3.2.10. <automata> element .................................. 8
       3.2.11. <class> element .................................... 8
       3.2.12. <duplex> element ................................... 9
       3.2.13. <description> element ............................. 9
       3.2.14. <event-packages> element .......................... 10
       3.2.15. <priority> element ................................ 10
       3.2.16. <methods> element ................................ 11
       3.2.17. <extensions> element .............................. 11
       3.2.18. <schemes> element ................................ 12
       3.2.19. <actor> element .................................... 12
       3.2.20. <isfocus> element .................................. 13
       3.2.21. <languages> element ................................ 13
   3.3. Device capability element ............................... 14
       3.3.1. <devcaps> element .................................. 14
       3.3.2. <mobility> element .................................. 14
       3.3.3. <priority> element ................................ 15
       3.3.4. <description> element ............................. 16
4. Usage guidelines .............................................. 16
5. Examples ..................................................... 16
6. XML schema definitions ....................................... 17
7. IANA Considerations .......................................... 24
   7.1. URN sub-namespace registration for 'urn:ietf:params:xml:ns:pidf:caps' ................................................. 24
8. Security Considerations ....................................... 25
9. Acknowledgments ............................................. 25
10. References .................................................. 26
    10.1. Normative references ................................... 26
    10.2. Informative references ................................. 26
Authors’ Addresses .............................................. 28
1. Introduction

Interoperation of instant messaging and presence systems has been defined in IMPP Working Group (WG). The WG has defined "A model for Presence and Instant Messaging" in RFC2778 [2] and also requirements for protocols implementing such a system in RFC2779 [3]. Common presence (CPP) [5] and common profile for instant messaging (CPIM) [13] defines common operations and formats which all presence and instant messaging services must agree upon so that basic interoperability would be possible. The actual base format for the presence is defined in the Presence Information Document Format (PIDF) [4]. The PIDF has been designed to reduce the need for gatewaying and to allow end-to-end security of presence data. It has taken very minimalistic approach to support such operations. In order to make the PIDF usable by different presence applications, these applications usually must extend the basic PIDF by standard XML mechanisms as defined in the RFC3863 [4].

The aim of this memo is to introduce a SIP specific extension mechanism to the PIDF that conveys the same SIP media feature tags as described in RFC3840 [6]. With this extension presence applications based on SIP can have richer and more usable presence data compared to the baseline PIDF.

1.1. Motivation

The PIDF [4] defines a <contact> element which may appear once inside every <tuple> element. The content of the <contact> element encodes the CONTACT ADDRESS and CONTACT MEANS as defined in RFC2778 [2]. The <contact> element is defined to be a URI of any scheme. In some implementations the URI scheme can uniquely identify the service the tuple intends to describe (e.g. im: URI scheme usually represent Instant Messaging service). However, this may not be the case in all implementations. For example in SIP, a SIP URI scheme can represent different kinds of services. A SIP URI scheme can be used to contact voice services, video services, or messaging services. If it is not known by other means, it might be hard for applications processing the presence information document containing only a SIP URI contact addresses to know what particular service the tuple intends to describe. Also watchers receiving presence information would probably benefit for getting more descriptive information about what particular communication means or services are supported by the presentity.

RFC3840 [6] defines a set of extensions which allow callees to express preferences about request handling in SIP servers. The same information can provide value also to presence watchers so that they can make more rational decisions on how a presentity should be
contacted if a presence document would contained similar information.

1.2. Scope

This document defines extensions to the PIDF which enables presence implementations based on SIP to utilize similar information in a presence information document as what RFC3840 [6] defines.

This extension does not replace media negotiation mechanisms defined for SIP (e.g. SDP [17]). This extension is only aimed for presentities to give watchers hints about the presentity's preferences, willingness and capabilities to communicate before watchers initiate a communication with the presentity.

2. Conventions

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant implementations.

This memo makes use of the vocabulary defined in RFC2778 [2], and in RFC3863 [4].

3. Extension for "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)" in PIDF documents

This section presents all extension documents, their elements, their values, and semantics. This section also describes how this extension can be further extended.

This extension is intended to be used within a PIDF [4] and that particular usage is described here. This extension may also be used with other XML documents if appropriate.

3.1. Overview of operation

This document defines how the features presented in RFC3840 [6] can be provided as a part of presence data. Additionally, this memo includes the "type" feature tag [7], "message" media type feature tag [18] and the "language" feature tag [8] definitions. Bringing these features in the PIDF means mapping them to an XML formatted structure.

The presence data model [11] defines presence data consisting of three types of data elements: person, service, and device. This memo
follows this model so that one XML document is defined to describe device capabilities and another one to describe service capabilities.

The namespace URIs for elements defined by this document are URNs using the namespace identifier ‘ietf’ defined by RFC2648 [9] and extended by RFC3688 [10].

When these extension namespaces are congregated with the PIDF document, the combined document MUST follow the same general formatting rules as specified in Section 4.1 of the RFC3863 [4].

3.2. Service capability element

Elements belonging to service capabilities are used to describe characteristics of a service. All elements defined in this section describe dynamic data about the service. This element SHOULD be located in PIDF document as a child element of urn:ietf:params:xml:ns:pidf namespace <tuple> [4] element.

Namespace identifier for this element is:

urn:ietf:params:xml:ns:pidf:caps

3.2.1. <servcaps> element

The root element of service capabilities is <servcaps>. The root element has to be always present. This element can contain following elements: <audio>, <application>, <control>, <video>, <text>, <message>, <type>, <automata>, <class>, <duplex>, <description>, <event-packages>, <priority>, <methods>, <extensions>, <schemes>, <actor>, <isfocus>, and <languages> followed by any number of optional extension elements from other namespaces.

A <servcaps> element does not have any attributes.

3.2.2. <audio> element

The <audio> element indicates that the service supports audio as a streaming media type as defined in RFC3840 [6].

The <audio> element is a boolean type and does not have any attributes. The value ‘true’ indicates that service supports audio media type and the value ‘false’ indicates that service does not support audio media type.

3.2.3. <application> element

The <application> element indicates that the service supports
application as a streaming media type as defined in RFC3840 [6].

The <application> element is a boolean type and does not have any attributes. The value ‘true’ indicates that service supports application media type and the value ‘false’ indicates that service does not support application media type.

3.2.4. <data> element

The <data> element indicates that the service supports data as a streaming media type as defined in RFC3840 [6].

The <data> element is a boolean type and does not have any attributes. The value ‘true’ indicates that service supports data media type and the value ‘false’ indicates that service does not support data media type.

3.2.5. <control> element

The <control> element indicates that the service supports control as a streaming media type as defined in RFC3840 [6].

The <control> element is a boolean type and does not have any attributes. The value ‘true’ indicates that service supports control media type and the value ‘false’ indicates that service does not support control media type.

3.2.6. <video> element

The <video> element indicates that the service supports video as a streaming media type as defined in RFC3840 [6].

The <video> element is a boolean type and does not have any attributes. The value ‘true’ indicates that service supports video media type and the value ‘false’ indicates that service does not support video media type.

3.2.7. <text> element

The <text> element indicates that the service supports text as a streaming media type as defined in RFC3840 [6].

The <text> element is a boolean type and does not have any attributes. The value ‘true’ indicates that service supports text media type and the value ‘false’ indicates that service does not support text media type.
3.2.8. <message> element

The <message> element indicates that the service supports message as a streaming media type as defined in [18].

The <message> element is a boolean type and does not have any attributes. The value 'true' indicates that service supports message media type and the value 'false' indicates that service does not support message media type.

3.2.9. <type> element

The <type> element indicates a MIME media content type (i.e. that appears in a ‘Content-type:’ header of the corresponding MIME-formatted data) as defined in RFC2913 [7].

The <type> element is a string type and does not have any attributes. It MUST be a string of the form "type/subtype", where 'type' and 'subtype' are defined by the MIME specification [19]. Only lower-case letters SHOULD be used.

3.2.10. <automata> element

The <automata> element indicates whether the service represents an automata (such as a voicemail server, conference server, IVR, or recording device) or a human as defined in RFC3840 [6].

The <automata> element is a boolean type and does not have any attributes. The value 'true' indicates that the service represents an automata and the value 'false' indicates that it represents a human.

3.2.11. <class> element

The <class> element indicates the setting, business or personal, in which a communications service is used as defined in RFC3840 [6].

The <class> element can contain two elements: <supported> and <notsupported>. All classes that are supported by the service are listed under <supported> element and all classes that are not supported by the service are listed under <notsupported> element.

<supported> and <notsupported> elements can contain <business> and <personal> elements followed by any number of optional extension elements from other namespaces. The semantics of business and personal are defined in RFC3840 [6] as:
o  <business>: The service is used for business communications.

o  <personal>: The service is used for personal communications.

Any value that is register to IANA to SIP Media Feature Tag Registration Tree as sip.class media feature tag can be used as a value of an extension element. If appropriate value is not registered it SHOULD be registed as defined in RFC3840 [6].

3.2.12.  <duplex> element

The <duplex> element lists whether a communications service can simultaneously send and receive media ("full"), alternate between sending and receiving ("half"), can only receive ("receive-only") or only send ("send-only") as defined in RFC3840 [6].

The <duplex> element can contain two elements: <supported> and <notsupported>. All duplex modes that are supported by the service are listed under <supported> element and all duplex modes that are not supported by the service are listed under <notsupported> element.

<supported> and <notsupported> elements can contain <full>, <half>, <receive-only> and <send-only> elements followed by any number of optional extension elements from other namespaces. The semantics of these elements are defined in RFC3840 [6] as:

o  <full>: The service can simultaneously send and receive media.

o  <half>: The service can alternate between sending and receiving media.

o  <receive-only>: The service can only receive media.

o  <send-only>: The service can only send media.

Any value that is register to IANA to SIP Media Feature Tag Registration Tree as sip.duplex media feature tag can be used as a value of an extension element. If appropriate value is not registered it SHOULD be registed as defined in RFC3840 [6].

3.2.13.  <description> element

The <description> element provides a textual description of the service as defined in RFC3840 [6].

The <description> element is of string type and does not have any attributes.
3.2.14.  <event-packages> element

The <event-packages> element lists the event packages supported by a service.

The <event-packages> element can contain two elements: <supported> and <notsupported>. All event packages that are supported by the service are listed under <supported> element and all event packages that are not supported by the service are listed under <notsupported> element.

<supported> and <notsupported> elements can contain <presence>, <message-summary>, <reg>, <refer>, <winfo>, <spirits-user-prof>, <spirits-user-prof>, <spirits-INDPs>, and <Siemens-RTP-Stats> followed by any number of optional extension elements from other namespaces.

Any value that is register to IANA to SIP Event types namespace registry can be used as a value of an extension element.

3.2.15.  <priority> element

The <priority> element indicates the call priorities the service is willing to handle.

The <priority> element can contain two elements: <supported> and <notsupported>. All priority values that are supported by the service are listed under <supported> element and all priority values that are not supported by the service are listed under <notsupported> element.

The <supported> and <notsupported> elements can contain any number of <lowerthan>, <higherthan>, <equals> and <range> elements followed by any number of optional extension elements from other namespaces.

3.2.15.1.  <lowerthan> element

The <lowerthan> element has a single attribute called "maxvalue". The "maxvalue" attribute is used to give the highest priority value that the service is willing to support. All values equal and below that value are supported.

3.2.15.2.  <higherthan> element

The <higherthan> element has a single attribute called "minvalue". The "minvalue" attribute is used to give the lowest priority value that the service is willing to support. All values equal and above that value are supported.
3.2.15.3.  <equals> element

The <equals> element is used to indicate the exact priority value that the service is willing to handle. The <equals> element has a single attribute called "value". Value of the "value" attribute is used to indicate the exact supported priority value.

3.2.15.4.  <range> element

The <range> element is used to indicate the priority range that the service is willing to handle. The <range> element has two attributes called "min" and "max". Value of the "min" attribute indicates lowest priority value supported by the service and the value of the "max" attribute indicates the highest priority value supported by the service.

3.2.16.  <methods> element

The <methods> element indicates the SIP methods supported by a service. In this case, "supported" means that the service can receive requests with this method. In that sense, it has the same connotation as the Allow header field as defined in RFC3840 [6].

The <methods> element can contain two elements: <supported> and <notsupported>. All methods that are supported by the service are listed under <supported> element and all methods that are not supported by the service are listed under <notsupported> element.

The <supported> and <notsupported> elements can contain <ACK>, <BYE>, <CANCEL>, <INFO>, <INVITE>, <MESSAGE>, <NOTIFY>, <OPTIONS>, <PRACK>, <PUBLISH>, <REFER>, <REGISTER>, <SUBSCRIBE>, and <UPDATE> elements followed by any number of optional extension elements from other namespaces.

Any value that is defined in IANA SIP parameters registry methods table can be used as a value of an extension element.

3.2.17.  <extensions> element

The <extensions> element is a list of SIP extensions (each of which is defined by an option-tag registered with IANA) that are understood by the service. Understood, in this context, means that the option tag would be included in a Supported header field in a request as defined in RFC3840 [6].

The <extensions> element can contain two elements: <supported> and <notsupported>. All extensions that are supported by the service are listed under <supported> element and all extensions that are not
supported by the service are listed under <notsupported> element.

The <supported> and <notsupported> elements can contain <rel100>, <join>, <path>, <precondition>, <pref>, <privacy>, <replaces>, and <sec-agree> elements followed by any number of optional extension elements from other namespaces.

Any value that is defined in IANA SIP parameters registry option tags table can be used as a value of an extension element.

3.2.18. <schemes> element

The <schemes> element provides the set of URI schemes that are supported by a service. Supported implies, for example, that the service would know how to handle a URI of that scheme in the Contact header field of a redirect response as defined in [6].

The <schemes> element can contain two elements: <supported> and <notsupported>. All schemes that are supported by the service are listed under <supported> element and all schemes that are not supported by the service are listed under <notsupported> element.

The <supported> and <notsupported> elements can contain any number of <s> elements which can be used to describe individual schemes supported by the service.

3.2.18.1. <s> element

The <s> element is of string type and it is used to describe individual scheme supported by the service. Values that can be used here are scheme names that are registered to IANA URI scheme registry.

3.2.19. <actor> element

The <actor> element indicates the type of entity that is available at this URI as defined in RFC3840 [6].

The <actor> element can contain two elements: <supported> and <notsupported>. All actor types that are supported by the service are listed under <supported> element and all actor types that are not supported by the service are listed under <notsupported> element.

The <supported> and <notsupported> elements can contain <principal>, <attendant>, <msg-taker>, and <information> elements followed by any number of optional extension elements from other namespaces.

The semantics of these elements are defined in RFC3840 [6] as:
o  <principal>: The service provides communication with the principal that is associated with the service. Often this will be a specific human being, but it can be an automata (for example, when calling a voice portal).

o  <attendant>: The service provides communication with an automata or person that will act as an intermediary in contacting the principal associated with the service, or a substitute.

o  <msg-taker>: The service provides communication with an automata or person that will take messages and deliver them to the principal.

o  <information>: The service provides communication with an automata or person that will provide information about the principal.

Any value that is register to IANA to SIP Media Feature Tag Registration Tree as sip.actor media feature tag can be used as a value of an extension element. If appropriate value is not registered it SHOULD be registered as defined in RFC3840 [6].

3.2.20.  <isfocus> element

The <isfocus> element indicates that the service is a conference server, also known as a focus as defined in RFC3840 [6].

The <isfocus> element is of boolean type and does not have any attributes.

3.2.21.  <languages> element

The <languages> element indicates the ability to display particular human languages as defined in RFC2987 [8].

The <languages> element can contain two elements: <supported> and <notsupported>. All languages that are supported by the service are listed under <supported> element and all languages that are not supported by the service are listed under <notsupported> element.

<supported> and <notsupported> elements can contain any number of <l> elements which can be used to describe individual languages supported by the service.

3.2.21.1.  <l> element

The <l> element is of string type and it is used to describe individual language supported by the service. Values that can be used here are language names that are registered to IANA as per
3.3. Device capability element


Namespace identifier for this element is urn:ietf:params:xml:ns:pidf:caps.

3.3.1. <devcaps> element

The root element of device capabilities is <devcaps>. The root element has to be always present. This element can contain following elements: <mobility>, <priority>, and <description> followed by any number of optional extension elements from other namespaces.

A <devcaps> element does not have any attributes.

3.3.2. <mobility> element

The <mobility> element indicates whether the device is fixed (meaning that it is associated with a fixed point of contact with the network), or mobile (meaning that it is not associated with a fixed point or contact). Note that cordless phones are fixed, not mobile, based on this definition as defined in RFC3840 [6].

The <mobility> element can contain two elements: <supported> and <notsupported>. All mobility modes that are supported by the device are listed under <supported> element and all mobility modes that are not supported by the device are listed under <notsupported> element.

The <supported> and <notsupported> elements can contain <fixed> and <mobile> elements followed by any number of optional extension elements from other namespaces.

The semantics of these elements are defined in RFC3840 [6] as:

- <fixed>: The device is stationary.
- <mobile>: The device can move around with the user.

Any value that is register to IANA to SIP Media Feature Tag Registration Tree as sip.mobility media feature tag can be used as a
value of an extension element. If appropriate value is not registered it SHOULD be registered as defined in RFC3840 [6].

3.3.3. <priority> element

The <priority> element indicates the call priorities the device is willing to handle.

The <priority> element can contain two elements: <supported> and <notsupported>. All priority values that are supported by the device are listed under <supported> element and all priority values that are not supported by the device are listed under <notsupported> element.

The <supported> and <notsupported> elements can contain any number of <lowerthan>, <higherthan>, <equals> and <range> elements followed by any number of optional extension elements from other namespaces.

3.3.3.1. <lowerthan> element

The <lowerthan> element has a single attribute called "maxvalue". The "maxvalue" attribute is used to give the highest priority value that the device is willing to support. All values equal and below that value are supported.

3.3.3.2. <higherthan> element

The <higherthan> element has a single attribute called "minvalue". The "minvalue" attribute is used to give the lowest priority value that the device is willing to support. All values equal and above that value are supported.

3.3.3.3. <equals> element

The <equals> element is used to indicate the exact priority value that the device is willing to handle. The <equals> element has a single attribute called "value". Value of the "value" attribute is used to indicate the exact supported priority value.

3.3.3.4. <range> element

The <range> element is used to indicate the priority range that the device is willing to handle. The <range> element has two attributes called "min" and "max". Value of the "min" attribute indicates lowest priority value supported by the device and the value of the "max" attribute indicates the highest priority value supported by the device.
3.3.4.  <description> element

The <description> element provides a textual description of the device as defined in RFC3840 [6].

The <description> element is of string type and does not have any attributes.

4. Usage guidelines

In RFC3840 [6] it is recommended that a UA provides complete information in its contact predicate. However, it may be that presentity is not willing to publish presence information which would be consistent with actual device or service capabilities (e.g. presentity may not want to indicate that he/she supports voice when the service actually is able to support it). Also authorization rules or policies in presence server may limit or modify the published presence information in a way that all published presence information may not end up to all possible watchers.

It is RECOMMENDED that Presence User Agents (PUAs) using this extension provide as complete presence information as they can. PUAs can indicate all the explicitly supported capabilities using the <supported> element and all the explicitly not supported capabilities using the <notsupported> element, where appropriate.

However, it is not mandated that this presence information should be consistent with actual device capabilities. Also, watchers should not expect that the presence information represented by this extension fully represents the actual presentity’s device capabilities.

5. Examples

<?xml version="1.0" encoding="UTF-8"?>
<pres xmlns="urn:ietf:params:xml:ns:pidf"
     xmlns:caps="urn:ietf:params:xml:ns:pidf:caps"
     entity="pres:someone@example.com">
   <tuple id="joi9877866786ua9">
     <status>
       <basic>open</basic>
     </status>
     <caps:servcaps>
       <caps:audio>true</caps:audio>
     </caps:servcaps>
   </tuple>
</pres>
<caps:description>
  Example service
</caps:description>
<caps:duplex>
  <caps:full/>
</caps:duplex>
<caps:message>true</caps:message>
<caps:methods>
  <caps:ACK/>
  <caps:BYE/>
  <caps:INVITE/>
</caps:methods>
<caps:priority>
  <caps:lowerthan maxvalue="10"/>
</caps:priority>
<caps:schemes>
  <caps:s>sip</caps:s>
</caps:schemes>
<caps:video>false</caps:video>
</caps:servcaps>
<contact>sip:someone@example.com</contact>
</tuple>
<mod:device id="hgt67">
  <caps:devcaps>
    <caps:mobility>
      <caps:mobile/>
    </caps:mobility>
  </caps:devcaps>
  <mod:deviceID>urn:uuid:d27459b7-8213-4395-aa77-ed859a3e5b3a</mod:deviceID>
</mod:device>
</presence>

6. XML schema definitions

This section gives the XML schema definitions for the extensions defined in this document. Namespace identifier for this schema is urn:ietf:params:xml:ns:pidf:caps.
<?xml version="1.0" encoding="UTF-8"?>
  <!-- This import brings in the XML language attribute xml:lang-->
  <!-- ROOT -->
  <xs:element name="servcaps" type="tns:servcapstype">  
    <xs:sequence>
      <xs:element name="actor" type="tns:actortype" minOccurs="0"/>
      <xs:element name="application" type="tns:applicationtype" minOccurs="0"/>
      <xs:element name="audio" type="tns:audiotype" minOccurs="0"/>
      <xs:element name="automata" type="tns:automatatype" minOccurs="0"/>
      <xs:element name="class" type="tns:classtype" minOccurs="0"/>
      <xs:element name="control" type="tns:controltype" minOccurs="0"/>
      <xs:element name="data" type="tns:datatype" minOccurs="0"/>
      <xs:element name="description" type="tns:descriptiontype" minOccurs="0"/>
      <xs:element name="duplex" type="tns:duplextype" minOccurs="0"/>
      <xs:element name="extensions" type="tns:extensionstype" minOccurs="0"/>
      <xs:element name="event-packages" type="tns:event-packagestype" minOccurs="0"/>
      <xs:element name="isfocus" type="tns:isfocustype" minOccurs="0"/>
      <xs:element name="message" type="tns:messagetype" minOccurs="0"/>
      <xs:element name="methods" type="tns:methodstype" minOccurs="0"/>
      <xs:element name="languages" type="tns:languagestype" minOccurs="0"/>
      <xs:element name="priority" type="tns:prioritytype" minOccurs="0"/>
      <xs:element name="schemes" type="tns:schemestype" minOccurs="0"/>
      <xs:element name="text" type="tns:texttype" minOccurs="0"/>
      <xs:element name="type" type="tns:typetype" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="video" type="tns:videotype" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="devcaps" type="tns:devcaps">  
    <xs:sequence>
      <xs:element name="description" type="tns:descriptiontype" minOccurs="0"/>
      <xs:element name="mobility" type="tns:mobilitytype" minOccurs="0"/>
      <xs:element name="priority" type="tns:prioritytype" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- AUDIO -->
  <xs:simpleType name="audiotype">
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:schema>
<!-- APPLICATION -->
<xs:simpleType name="applicationtype">
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>

<!-- DATA -->
<xs:simpleType name="datatype">
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>

<!-- CONTROL -->
<xs:simpleType name="controltype">
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>

<!-- VIDEO -->
<xs:simpleType name="videotype">
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>

<!-- TEXT -->
<xs:simpleType name="texttype">
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>

<!-- MESSAGE -->
<xs:simpleType name="messagetype">
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>

<!-- TYPE -->
<xs:simpleType name="typetype">
  <xs:restriction base="xs:string"/>
</xs:simpleType>

<!-- AUTOMATA -->
<xs:simpleType name="automatatype">
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>

<!-- CLASS -->
<xs:complexType name="classtype">
  <xs:sequence>
    <xs:element name="supported" type="tns:classtypes" minOccurs="0"/>
    <xs:element name="notsupported" type="tns:classtypes" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="classtypes">
</xs:complexType>
<xs:sequence>
  <xs:element name="business" type="xs:string" minOccurs="0"/>
  <xs:element name="personal" type="xs:string" minOccurs="0"/>
  <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<!-- DUPLEX -->
<xs:complexType name="duplextype">
  <xs:sequence>
    <xs:element name="supported" type="tns:duplextypes" minOccurs="0"/>
    <xs:element name="notsupported" type="tns:duplextypes" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="duplextypes">
  <xs:sequence>
    <xs:element name="full" type="xs:string" minOccurs="0"/>
    <xs:element name="half" type="xs:string" minOccurs="0"/>
    <xs:element name="receive-only" type="xs:string" minOccurs="0"/>
    <xs:element name="send-only" type="xs:string" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- DESCRIPTION -->
<xs:simpleType name="descriptiontype">
  <xs:restriction base="xs:string"/>
</xs:simpleType>

<!-- EVENT-PACKAGES -->
<xs:complexType name="event-packagestype">
  <xs:sequence>
    <xs:element name="supported" type="tns:eventtypes" minOccurs="0"/>
    <xs:element name="notsupported" type="tns:eventtypes" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="eventtypes">
  <xs:sequence>
    <xs:element name="message-summary" type="xs:string" minOccurs="0"/>
    <xs:element name="presence" type="xs:string" minOccurs="0"/>
    <xs:element name="refer" type="xs:string" minOccurs="0"/>
    <xs:element name="reg" type="xs:string" minOccurs="0"/>
    <xs:element name="Siemens-RTP-Stats" type="xs:string" minOccurs="0"/>
    <xs:element name="spirits-INDPs" type="xs:string" minOccurs="0"/>
    <xs:element name="spirits-user-prof" type="xs:string" minOccurs="0"/>
    <xs:element name="winfo" type="xs:string" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- PRIORITY -->
<xs:complexType name="prioritytype">
  <xs:sequence>
    <xs:element name="supported" type="tns:prioritytypes" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="notsupported" type="tns:prioritytypes" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- METHODS -->
<xs:complexType name="methodstype">
  <xs:sequence>
    <xs:element name="supported" type="tns:methodtypes" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="notsupported" type="tns:methodtypes" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- EQUALS -->
<xs:complexType name="equalstype">
  <xs:attribute name="value" type="xs:integer" use="required"/>
</xs:complexType>

<!-- HIGHER THAN -->
<xs:complexType name="higherthantype">
  <xs:attribute name="minvalue" type="xs:integer" use="required"/>
</xs:complexType>

<!-- LOWER THAN -->
<xs:complexType name="lowerthantype">
  <xs:attribute name="maxvalue" type="xs:integer" use="required"/>
</xs:complexType>

<!-- RANGE -->
<xs:complexType name="rangetype">
  <xs:attribute name="minvalue" type="xs:integer" use="required"/>
  <xs:attribute name="maxvalue" type="xs:integer" use="required"/>
</xs:complexType>
<!-- EXTENSIONS -->
<xs:complexType name="extensionstype">
  <xs:sequence>
    <xs:element name="supported" type="tns:extensiontypes" minOccurs="0"/>
    <xs:element name="notsupported" type="tns:extensiontypes" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- SCHEMES -->
<xs:complexType name="schemestype">
  <xs:sequence>
    <xs:element name="supported" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="s" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="notsupported" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="s" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="actortype">
  <xs:sequence>
    <xs:element name="supported" type="tns:actortypes" minOccurs="0"/>
    <xs:element name="notsupported" type="tns:actortypes" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="actortypes">
  <xs:sequence>
    <xs:element name="attendant" type="xs:string" minOccurs="0"/>
    <xs:element name="information" type="xs:string" minOccurs="0"/>
    <xs:element name="msg-taker" type="xs:string" minOccurs="0"/>
    <xs:element name="principal" type="xs:string" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="isfocustype">
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>

<xs:complexType name="languagestype">
  <xs:sequence>
    <xs:element name="supported" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="notsupported" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="mobilitytype">
  <xs:element name="supported" minOccurs="0"/>
  <xs:element name="notsupported" minOccurs="0"/>
</xs:complexType>

<xs:complexType name="mobilitytype">
  <xs:element name="supported" minOccurs="0"/>
  <xs:element name="notsupported" minOccurs="0"/>
</xs:complexType>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="l" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
This memo calls for IANA to register one new XML namespace URNs as defined in RFC3688 [10].

7.1. URN sub-namespace registration for

    urn:ietf:params:xml:ns:pidf:caps


Description:
This is the XML namespace for XML elements defined by [[[RFCXXXX]]] to describe service and device capabilities in application/pidf+xml content type.

Registrant Contact:
IETF, SIMPLE working group, <simple@ietf.org>
Mikko Lonnfors, <mikko.lonnfors@nokia.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>Namespace for PIDF user agent capability extension</title>
</head>
<body>
<h1>Namespace for PIDF service capability extension</h1>
<h2>urn:ietf:params:xml:ns:pidf:caps</h2>
<p>See <a href="[[URL of published RFC]]">RFCXXXX</a>.</p>
</body>
</html>
END

8. Security Considerations


Because presence is very privacy-sensitive information, the transport protocol for the presence information SHOULD have capabilities to protect protocol messages from possible threats, such as eavesdropping, corruption, tamper and replay attacks. The protocols SHOULD be able to use security mechanisms which are standardized or being standardized in IETF. However, it depends on the actual transport protocols which security mechanisms should be used, and it is beyond the scope of this memo.

9. Acknowledgments

Authors of this document would like to thank following people for their contributions and valuable comments: Paul Kyzivat, Jonathan Rosenberg, Markus Isomaki, Eva Leppanen, Miguel Garcia, and Hisham Khartabil.

10. References
10.1. Normative references


10.2. Informative references


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


[18] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
Extensions (MIME) part two: Media Types", RFC 2046,
November 1996.

[19] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
Extensions (MIME) part one: Format of Internet Message Bodies",
RFC 2045, November 1996.


Resource Identifiers (URI): Generic Syntax", RFC 2236,
August 1998.
Authors’ Addresses

Mikko Lonnfors
Nokia
Itamerenkatu 11-13 00180
Helsinki
Finland

Phone: +358 71 8008000
Email: mikko.lonnfors@nokia.com

Krisztian Kiss
Nokia
12278 Scripps Summit Drive
San Diego, CA 92131
USA

Phone: +18583357341
Email: krisztian.kiss@nokia.com
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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