Network Working Group
Request for Comments: 5196
Category: Standards Track

M. Lonnfors
K. Kiss
Nokia
September 2008

Session Initiation Protocol (SIP) User Agent Capability Extension to Presence Information Data Format (PIDF)

Status of This Memo

Abstract

Presence Information Data Format (PIDF) defines a common presence data format for Common Profile for Presence (CPP) compliant presence protocols. This memo defines a PIDF extension to represent SIP User Agent capabilities.

Table of Contents

1. Introduction

Common Profile for Presence (CPP) [RFC3859] and Common Profile for Instant Messaging (CPIM) [RFC3860] define common operations and formats that all presence and instant messaging services must agree upon so that basic interoperability is possible. The actual base format for the presence is defined in the Presence Information Document Format (PIDF) [RFC3863]. The PIDF has been designed to reduce the need for gatewaying and to allow end-to-end security of presence information. It has taken a 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 PIDF [RFC3863].

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]. With this extension, presence applications based on SIP can have richer and more usable presence information compared to the baseline PIDF.

1.1. Motivation

The PIDF [RFC3863] defines a <contact> element that may appear once inside every <tuple> element. The content of the <contact> element encodes the CONTACT ADDRESS and CONTACT MEANS as defined in [RFC2778]. 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 represents 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 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 from getting more descriptive information about what particular communication means or services are supported by the presentity.

The User Agent Capabilities extension [RFC3840] defines a set of extensions that allow user agents to express preferences about request handling in SIP servers. The same information can provide value to watchers as well so that they can make more rational decisions on how a presentity should be contacted if a presence document contained this information.

1.2. Scope

This document defines a PIDF extension, which enables SIP presence implementations to represent User Agent Capabilities [RFC3840] within presence information.

This extension does not replace media negotiation mechanisms defined for SIP (e.g., SDP [RFC4566]). The purpose of this extension is for a presentity to give watchers hints about the presentity’s preferences, willingness, and capabilities to communicate before watchers initiate communication with the presentity.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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

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

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

This extension is intended to be used within the PIDF [RFC3863] 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] can be provided as part of presence information. Additionally, this memo includes the "type" feature tag [RFC2913], "message" media type feature tag [RFC4569], and the "language" feature tag [RFC4646] definitions. Adding these features to the PIDF means mapping them to an XML formatted structure.

The presence data model [RFC4479] defines presence information consisting of three types of data elements: person, service, and device. This memo follows this model so that one XML extension 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] and extended by [RFC3688].

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 [RFC3863].

3.2. Service capabilities

Elements belonging to service capabilities are used to describe dynamic characteristics of a service. These capabilities are enclosed within the <servcaps> element which SHOULD be located in the PIDF document as a child element of urn:ietf:params:xml:ns:pidf namespace <tuple> [RFC3863] element.

The namespace identifier for these elements is:

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

3.2.1. <servcaps> Element

The root element of service capabilities is <servcaps>. The root element always has to be present. This element can contain the following child elements: <audio>, <application>, <data>, <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 can contain any number of optional extension attributes from other namespaces.

3.2.2. <audio> Element

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

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].

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].

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].

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].

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].

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 messaging as a streaming media type as defined in [RFC4569].

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].

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 [RFC2045]. Only lowercase letters SHOULD be used.

3.2.10. <automata> Element

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

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

The <class> element can contain two elements: <supported> and <notsupported>. Classes that are supported by the service can be listed under the <supported> element, and classes that are not supported by the service can be listed under the <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] as:

- <business>: The service is used for business communications.

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

Any value that is registered with IANA for the SIP media feature tag registration tree as a sip.class media feature tag can be used as a value of an extension element. If the appropriate value is not registered, it SHOULD be registered as defined in [RFC3840].

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"), only receive ("receive-only"), or only send ("send-only") as defined in [RFC3840].

The <duplex> element can contain two elements: <supported> and <notsupported>. Duplex modes that are supported by the service can be listed under the <supported> element, and duplex modes that are not supported by the service can be listed under the <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] as:

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

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

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

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

Any value that is registered with IANA for the SIP media feature tag registration tree as a sip.duplex media feature tag can be used as a value of an extension element. If the appropriate value is not registered, it SHOULD be registered as defined in [RFC3840].

3.2.13. <description> Element

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

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

The <description> element SHOULD be labeled with the ’xml:lang’ attribute to indicate its language and script. The specification allows multiple occurrences of this elements so that the presentity can convey <description> elements in multiple scripts and languages. If no ’xml:lang’ attribute is provided, the default value is "i-default" as defined in [RFC2277].

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>. Event packages that are supported by the service can be listed under the <supported> element, and event packages that are not supported by the service can be listed under the <notsupported> element.

The <supported> and <notsupported> elements can contain any values from the IANA SIP event types namespace registry followed by any number of optional extension elements from other namespaces. As of this writing, the IANA SIP event types namespace registry includes the following packages: <conference>, <dialog>, <kpml>, <message-summary>, <poc-settings>, <presence>, <reg>, <refer>, <Siemens-RTP-Stats>, <spirits-INDPs>, <spirits-user-prof>, and <winfo>.

3.2.15. <priority> Element

The <priority> element indicates the call priorities the service is willing to handle as defined in [RFC3840].

The <priority> element can contain two elements: <supported> and <notsupported>. Priority values that are supported by the service can be listed under the <supported> element, and priority values that are not supported by the service can be listed under the <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". 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 "minvalue" and "maxvalue". The value of the "minvalue" attribute indicates the lowest priority value supported by the service, and the value of the "maxvalue" 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].

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

The <supported> and <notsupported> elements can contain any values from the methods table of the IANA SIP parameters registry table followed by any number of optional extension elements from other namespaces. As of this writing, the IANA SIP parameters registry includes the following methods:<ACK>, <BYE>, <CANCEL>, <INFO>, <INVITE>, <MESSAGE>, <NOTIFY>, <OPTIONS>, <PRACK>, <PUBLISH>, <REFER>, <REGISTER>, <SUBSCRIBE>, and <UPDATE>.

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].

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

The <supported> and <notsupported> elements can contain any values from the option tags table of the IANA SIP parameters registry table followed by any number of optional extension elements from other namespaces. As of this writing, the IANA SIP parameters registry includes the following option tags: <rel100>, <early-session>, <eventlist>, <from-change>, <gruu>, <histinfo>, <join>, <norefersub>, <path>, <precondition>, <pref>, <privacy>, <recipient-list-invite>, <recipient-list-subscribe>, <replaces>, <resource-priority>, <sdp-anat>, <sec-agree>, <tdialog>, and <timer>.

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 [RFC3840].

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

<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 is used to describe an individual scheme supported by the service. Values that can be used here are scheme names that are registered to the 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].

The <actor> element can contain two elements: <supported> and <notsupported>. Actor types that are supported by the service can be listed under the <supported> element, and actor types that are not supported by the service can be listed under the <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] as:

- <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 automaton (for example, when calling a voice portal).

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

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

- <information>: The service provides communication with an automaton or a person that will provide information about the principal.

Any value that is registered with IANA for the SIP media feature tag registration tree as a sip.actor media feature tag can be used as a value of an extension element. If the appropriate value is not registered, it SHOULD be registered as defined in [RFC3840].

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].

The <isfocus> element is of boolean type and does not have any attributes. The value 'true' indicates that service is a conference server and the value 'false' indicates that service does not support conferencing.

3.2.21. <languages> Element

The <languages> element indicates the ability to display particular human languages as defined in [RFC4646].

The <languages> element can contain two elements: <supported> and <notsupported>. Languages that are supported by the service can be listed under the <supported> element, and languages that are not supported by the service can be listed under the <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 is used to describe an individual language supported by the service. Values that can be used here are language subtags that are registered to the IANA language subtag registry as per [RFC4646].

3.3. Device Capabilities

Elements belonging to device capabilities are used to describe dynamic characteristics of a device. These capabilities are enclosed within the <devcaps> element, which SHOULD be located in the PIDF document as a child element of the urn:ietf:params:xml:ns:pidf:data-model namespace <device> element [RFC4479].

The namespace identifier for these elements is urn:

ietf:params:xml:ns:pidf:caps

3.3.1. <devcaps> Element

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

A <devcaps> element can contain any number of optional extension attributes from other namespaces.

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 of contact). Note that cordless phones are fixed, not mobile, based on this definition as defined in [RFC3840].

The <mobility> element can contain two elements: <supported> and <notsupported>. Mobility modes that are supported by the device can be listed under the <supported> element and mobility modes that are not supported by the device can be listed under the <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] as:

- <fixed>: The device is stationary.

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

Any value that is registered with IANA to the SIP media feature tag registration tree as sip.mobility media feature tag can be used as a value of an extension element. If the appropriate value is not registered, it SHOULD be registered as defined in [RFC3840].

3.3.3. <description> Element

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

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

The <description> element SHOULD be labeled with the ’xml:lang’ attribute to indicate its language and script. The specification allows multiple occurrences of this element so that the presentity can convey <description> elements in multiple scripts and languages. If no ’xml:lang’ attribute is provided, the default value is "i-default" as defined in [RFC2277].

4. Usage Guidelines

The User Agent Capabilities extension [RFC3840] recommends that a UA provides complete information in its contact predicate. However, it may be that the presentity is not willing to publish presence information that 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). Authorization rules or policies in the presence server may limit or modify the presence information published by the presentity. Also, combining presence information from multiple sources may result in loss or mismatch of information.

It is RECOMMENDED that Presence User Agents (PUAs) using this extension provide as complete presence information as they can. If the PUA is publishing sensitive information using this extension, it SHOULD obtain permission from the presentity. PUAs can indicate the explicitly supported capabilities using the <supported> element, and the capabilities that are explicitly not supported using the <notsupported> element.

It is not mandated that presence information be consistent with actual service or device capabilities. However, it is in the presentity’s best interest to avoid publishing false presence information and provide accurate information to help minimize unsuccessful communication invitations. Otherwise, watchers may conclude that communication cannot be established with the presentity, but in reality it would be possible; or watchers may conclude that certain communication capabilities are available, but in reality a communication establishment attempt would fail using those capabilities. In any case, watchers should not expect the presence information represented by this extension to be fully aligned with the actual presentity’s service or device capabilities. As explained in Section 1.2, presence of this extension does not replace the use of SIP signaling for capability negotiation.

4.1. Use of <supported> and <notsupported> Elements

PUAs should add information under <supported> and <notsupported> elements only when they believe it may affect the decision making in the watcher’s end, i.e., information should be relevant and valuable for the watcher. Listing all possible information under <supported> and <notsupported> is rarely needed.

For example, if the PUA wants to advertise a message service that supports the MESSAGE method, it should add it under the <supported> element in the <methods> element. Even if the service does not support other methods, it is unlikely that listing all the methods not supported under the <notsupported> element would provide any value to the watcher.

In case of conflicting information, i.e., the same child element appears under the <supported> and <notsupported> elements with the same value, the watcher can safely assume that the listed capability is supported regardless of the inclusion of the capability under the <notsupported> element.

5. Examples

```
   <?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
         xmlns:caps="urn:ietf:params:xml:ns:pidf:caps"
         xmlns:mod="urn:ietf:params:xml:ns:pidf:data-model"
         entity="pres:someone@example.com">

     <tuple id="joi9877866786ua9">
       <status>
         <basic>open</basic>
       </status>
       <caps:servcaps>
         <caps:audio>true</caps:audio>
         <caps:description xml:lang="en">
            Example service
         </caps:description>
         <caps:description xml:lang="hu">
            Pe’lda szolga’ltata’s
         </caps:description>
         <caps:duplex>
           <caps:supported>
             <caps:full/>
           </caps:supported>
         </caps:duplex>
         <caps:message>true</caps:message>
         <caps:methods>
           <caps:supported>
             <caps:ACK/>
             <caps:BYE/>
             <caps:INVITE/>
             <caps:MESSAGE/>
           </caps:supported>
         </caps:methods>
         <caps:priority>
           <caps:supported>
             <caps:lowerthan maxvalue="10"/>
           </caps:supported>
         </caps:priority>
         <caps:schemes>
           <caps:supported>
             <caps:s>sip</caps:s>
           </caps:supported>
         </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:supported>
             <caps:mobile/>
           </caps:supported>
         </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. The namespace identifier for this schema is urn:ietf:params:xml:ns:pidf:caps.

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

<!-- This import brings in the XML language
     attribute xml:lang-->

 <xs:import namespace="http://www.w3.org/XML/1998/namespace"
  schemaLocation="http://www.w3.org/2001/xml.xsd"/>

<!-- ROOT -->
 <xs:element name="servcaps" type="tns:servcapstype"/>
 <xs:complexType name="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" maxOccurs="unbounded"/>
   <xs:element name="duplex" type="tns:duplextype"
    minOccurs="0"/>
   <xs:element name="event-packages" type="tns:event-packagestype"
    minOccurs="0"/>
   <xs:element name="extensions" type="tns:extensionstype"
    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:anyAttribute namespace="##any" processContents="lax"/>
 </xs:complexType>

 <xs:element name="devcaps" type="tns:devcaps"/>
 <xs:complexType name="devcaps">
  <xs:sequence>
   <xs:element name="description" type="tns:descriptiontype"
    minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="mobility" type="tns:mobilitytype"
    minOccurs="0"/>
   <xs:any namespace="##other" processContents="lax"
    minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 <xs:anyAttribute namespace="##any" processContents="lax"/>
 </xs:complexType>

 <!-- AUDIO -->
 <xs:simpleType name="audiotype">
  <xs:restriction base="xs:boolean"/>
 </xs:simpleType>

 <!-- 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: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:complexType name="descriptiontype">
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute ref="xml:lang"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <!-- 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="conference" type="xs:string"
    minOccurs="0"/>
   <xs:element name="dialog" type="xs:string"
    minOccurs="0"/>
   <xs:element name="kpml" type="xs:string"
    minOccurs="0"/>
   <xs:element name="message-summary" type="xs:string"
    minOccurs="0"/>
   <xs:element name="poc-settings" type="xs:string"
    minOccurs="0"/>
   <xs:element name="presence" type="xs:string"
    minOccurs="0"/>
   <xs:element name="reg" type="xs:string"
    minOccurs="0"/>
   <xs:element name="refer" 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"/>
   <xs:element name="notsupported" type="tns:prioritytypes"
    minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="prioritytypes">
  <xs:sequence>
   <xs:element name="equals" type="tns:equalstype"
    minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="higherhan" type="tns:higherthantype"
    minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="lowerthan" type="tns:lowerthantype"
    minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="range" type="tns:rangetype"
    minOccurs="0" maxOccurs="unbounded"/>
   <xs:any namespace="##other" processContents="lax"
    minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="lowerthantype">
  <xs:attribute name="maxvalue" type="xs:integer"
   use="required"/>
 </xs:complexType>
 <xs:complexType name="higherthantype">
  <xs:attribute name="minvalue" type="xs:integer"
   use="required"/>
 </xs:complexType>
 <xs:complexType name="equalstype">
  <xs:attribute name="value" type="xs:integer"
   use="required"/>
 </xs:complexType>
 <xs:complexType name="rangetype">
  <xs:attribute name="minvalue" type="xs:integer"
   use="required"/>
  <xs:attribute name="maxvalue" type="xs:integer"
   use="required"/>
 </xs:complexType>

 <!-- METHODS -->
 <xs:complexType name="methodstype">
  <xs:sequence>
```

```
    <xs:element name="supported" type="tns:methodtypes"
     minOccurs="0"/>
    <xs:element name="notsupported" type="tns:methodtypes"
     minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="methodtypes">
   <xs:sequence>
    <xs:element name="ACK" type="xs:string" minOccurs="0"/>
    <xs:element name="BYE" type="xs:string" minOccurs="0"/>
    <xs:element name="CANCEL" type="xs:string" minOccurs="0"/>
    <xs:element name="INFO" type="xs:string" minOccurs="0"/>
    <xs:element name="INVITE" type="xs:string" minOccurs="0"/>
    <xs:element name="MESSAGE" type="xs:string" minOccurs="0"/>
    <xs:element name="NOTIFY" type="xs:string" minOccurs="0"/>
    <xs:element name="OPTIONS" type="xs:string" minOccurs="0"/>
    <xs:element name="PRACK" type="xs:string" minOccurs="0"/>
    <xs:element name="PUBLISH" type="xs:string" minOccurs="0"/>
    <xs:element name="REFER" type="xs:string" minOccurs="0"/>
    <xs:element name="REGISTER" type="xs:string" minOccurs="0"/>
    <xs:element name="SUBSCRIBE" type="xs:string" minOccurs="0"/>
    <xs:element name="UPDATE" type="xs:string" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
     maxOccurs="unbounded"/>
   </xs:sequence>
  </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>
  <xs:complexType name="extensiontypes">
   <xs:sequence>
    <xs:element name="rel100" type="xs:string" minOccurs="0"/>
    <xs:element name="early-session" type="xs:string" minOccurs="0"/>
    <xs:element name="eventlist" type="xs:string" minOccurs="0"/>
    <xs:element name="from-change" type="xs:string" minOccurs="0"/>
    <xs:element name="gruu" type="xs:string" minOccurs="0"/>
    <xs:element name="hist-info" type="xs:string" minOccurs="0"/>
    <xs:element name="join" type="xs:string" minOccurs="0"/>
    <xs:element name="norefersub" type="xs:string" minOccurs="0"/>
    <xs:element name="path" type="xs:string" minOccurs="0"/>
    <xs:element name="precondition" type="xs:string" minOccurs="0"/>
    <xs:element name="pref" type="xs:string" minOccurs="0"/>
    <xs:element name="privacy" type="xs:string" minOccurs="0"/>
    <xs:element name="recipient-list-invite" type="xs:string"
     minOccurs="0"/>
    <xs:element name="recipient-list-subscribe" type="xs:string"
     minOccurs="0"/>
    <xs:element name="replaces" type="xs:string" minOccurs="0"/>
    <xs:element name="resource-priority" type="xs:string" minOccurs="0"/>
    <xs:element name="sdp-anat" type="xs:string" minOccurs="0"/>
    <xs:element name="sec-agree" type="xs:string" minOccurs="0"/>
    <xs:element name="tdialog" type="xs:string" minOccurs="0"/>
    <xs:element name="timer" type="xs:string" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax"
     minOccurs="0" maxOccurs="unbounded"/>
   </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>

  <!-- ACTOR -->
  <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>

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

 <!-- LANGUAGES -->
 <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>

 <!-- MOBILITY -->
 <xs:complexType name="mobilitytype">
  <xs:sequence>
   <xs:element name="supported" type="tns:mobilitytypes"
    minOccurs="0"/>
   <xs:element name="notsupported" type="tns:mobilitytypes"
    minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="mobilitytypes">
  <xs:sequence>
   <xs:element name="fixed" type="xs:string"
    minOccurs="0"/>
   <xs:element name="mobile" type="xs:string"
```

```
    minOccurs="0"/>
   <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

7. IANA Considerations

IANA has registered one new XML namespace URN and one schema as defined in [RFC3688].

7.1. URN Sub-Namespace Registration for 'urn:ietf:params:xml:ns:pidf:caps'

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

Description:
This is the XML namespace for XML elements defined by RFC 5196 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="http://www.rfc-editor.org/rfc/rfc5196.txt">RFC
         5196</a>.
       </p>
    </body>
    </html>
   END
```

7.2. Schema Registration for Schema 'urn:ietf:params:xml:schema:pidf:caps'

URI:
urn:ietf:params:xml:schema:pidf:caps

Registrant Contact:
IESG

XML:
See Section 6

8. Security Considerations

All security considerations specified in [RFC3859] and [RFC3863] apply to this document. Compared to PIDF [RFC3863], this presence document format may reveal additional information about user's service and device capabilities. Thus, the PUA SHOULD always obtain permission from the presentity when publishing sensitive information using this extension.

9. Acknowledgments

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

10. References

10.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) part one: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2913] Klyne, G., "MIME Content Types in Media Feature Expressions", RFC 2913, September 2000.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC3840] Schulzrinne, H., Rosenberg, J., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[RFC3859] Peterson, J., "Common Profile for Presence (CPP)", RFC 3859, August 2004.

[RFC3863] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, August 2004.

[RFC4479] Rosenberg, J., "A Data Model for Presence", RFC 4479, July 2006.

[RFC4646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

10.2. Informative References

[RFC2648] Moats, R., "A URN namespace for IETF documents", RFC 2648, August 1999.

[RFC2778] Day, M., Rosenberg, J., and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.

[RFC3860] Peterson, J., "Common Profile for Instant Messaging (CPIM)", RFC 3860, August 2004.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4569] Camarillo, G., "Internet Assigned Numbers Authority (IANA) Registration of the Message Media Feature Tag", RFC 4569, July 2006.

Authors' Addresses

Mikko Lonnfors
Nokia
P.O. Box 321
Helsinki
Finland

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

Krisztian Kiss
Nokia
313 Fairchild Dr
Mountain View, CA 94043
US

Phone: +1 650 391 5969
EMail: krisztian.kiss@nokia.com

Full Copyright Statement

Intellectual Property