Network Working Group
INTERNET-DRAFT

H. Sugano
S. Fujimoto
Fujitsu
G. Klyne
Nine by Nine
A. Bateman
VisionTech
W. Carr
Intel
J. Peterson
NeuStar

Expires: November 2003

May 2003

# Presence Information Data Format (PIDF)
<draft-ietf-impp-cpim-pidf-08.txt>

## Status of this Memo



## Copyright Notice

Abstract

This memo specifies the Common Profile for Presence (CPP) Presence Information Data Format (PIDF) as a common presence data format for CPP-compliant Presence protocols, and also defines a new media type "application/pidf+xml" to represent the XML MIME entity for PIDF.

Table of Content

## 1. Introduction

The Common Profiles for Instant Messaging (CPIM) [CPIM] and Presence (CPP) [CPP] specifications define a set of operations and parameters to achieve interoperability between different Instant Messaging and Presence protocols which meet RFC 2779 [RFC2779].

This memo further defines the Presence Information Data Format (PIDF) as a common presence data format for CPP-compliant presence protocols, allowing presence information to be transferred across CPP-compliant protocol boundaries without modification, with attendant benefits for security and performance.

The format specified in this memo defines the base presence format and extensibility required by RFC 2779. It defines a minimal set of presence status values defined by the IMPP Model document [RFC2778]. However, a presence application is able to define its own status values using the extensibility framework provided by this memo. Defining such extended status values is beyond the scope of this memo.

Note also that this memo defines only the format for a presence data payload and the extensibility framework for it. How the presence data is transferred within a specific protocol frame would be defined separately in a protocol specification.

### 1.1. Terminology and Conventions

This memo makes use of the vocabulary defined in the IMPP Model document [RFC2778]. Terms such as CLOSED, INSTANT MESSAGE, OPEN, PRESENCE SERVICE, PRESENTITY, WATCHER, and WATCHER USER AGENT in the memo are used in the same meaning as defined therein.

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2. Design Decisions

We have adopted the IMPP Model and Requirements documents [RFC2778, RFC2779] as the starting point of our discussion. The two RFCs contain a number of statements about presence information, which can be regarded as a basic set of constraints for the format design. Also, we took the minimalist approach to the design based on them. Starting from the minimal model, only the features that are necessary to solve particular problems have been included.





### 2.1. Minimal Model

This specification is based on the minimal model extracted from the IMPP Model and Requirements documents. The model consists of the following items. Each of them is accompanied with the corresponding RFCs and their section numbers as its grounds, e.g. (RFC2778:Sec.2.4) refers to Section 2.4 of RFC 2778.

(a) PRESENCE INFORMATION consists of one or more PRESENCE TUPLES, where a PRESENCE TUPLE consists of a STATUS, an optional COMMUNICATION ADDRESS, and optional OTHER PRESENCE MARKUP. Note that the CONTACT ADDRESS in a COMMUNICATIONS ADDRESS is understood in this document to refer only to a URI. (RFC2778:Sec.3)

(b) STATUS has at least the mutually-exclusive values OPEN and CLOSED, which have meaning for the acceptance of INSTANT MESSAGES, and may have meaning for other COMMUNICATION MEANS. There may be other values of STATUS that do not imply anything about INSTANT MESSAGE acceptance. These other values of STATUS may be combined with OPEN and CLOSED or they may be mutually-exclusive with those values. (RFC2778:Sec.3, RFC2779:Sec.4.4.1-

4.4.3)

(c) STATUS may consist of single or multiple values.(RFC2778:Sec.2.4)

(d) There must be a means of extending the common presence format to represent additional information not included in the common format. The extension and registration mechanisms must be defined for presence information schema, including new STATUS conditions and new forms for OTHER PRESENCE MARKUP. (RFC2779: Sec.3.1.4-3.1.5)

(e) The common presence format must include a means to uniquely identify the PRESENTITY whose PRESENCE INFORMATION is reported. (RFC2779:Sec.3.1.2)

(f) The common presence format must allow the PRESENTITY to secure presence information sent to a WATCHER. The format must allow integrity, confidentiality and authentication properties to be applied to presence information. (RFC2779:Sec5.2.1, 5.2.4, 5.3.1, 5.3.3)

## 2.2. Added Features

In addition to the minimal model described above, the format specified in this specification has the following features.





(a) Relative priorities of contact addresses are specifiable in order to allow the source of PRESENCE INFORMATION to tell the receiver (WATCHER USER AGENTS) its preference over multiple contact means.

(b) The presence format is able to contain the timestamp of the creation of the PRESENCE INFORMATION. The timestamp in the presence document lets the receiver know the time of the creation of the data even if the message containing it is delayed. It can also be used to detect a replay attack, independent of the underlying signature mechanism. Note that this mechanism does not assume any global time synchronization system for watchers and presentities (see Appendix A of RFC2779, 8.1.4 A7), but rather assumes that the minimum length of time that might pass before presence information is considered stale is long

enough that minor variations among system clocks will not lead to misjudgments of the freshness of presence information.

## 2.3. XML Encoding Decision

The Presence Information Data Format encodes presence information in XML (eXtensible Markup Language [XML]). Regarding the features of PRESENCE INFORMATION discussed above, such that it has a hierarchical structure and it should be fully extensible, XML is considered as the most desirable framework over other candidates such as vCard [RFC2426].

# 3. Overview of Presence Information Data Format

This section describes an overview of the presence data format defined in this memo.

## 3.1. The 'application/pidf+xml' Content Type

This memo defines a new content type "application/pidf+xml" for an XML MIME entity that contains presence information. This specification follows the recommendations and conventions described in [RFC3023], including the naming convention of the type ('+xml' suffix) and the usage of the 'charset' parameter.

Although it is defined as optional, use of the 'charset' parameter is RECOMMENDED. If the 'charset' parameter is not specified, conforming XML processors MUST follow the requirements in section 4.3.3 of [XML].





## 3.2. Presence Information Contents

This subsection outlines the information in an "application/pidf+xml" document. A full definition of the PIDF content is in Section 4.

- o PRESENTITY URL: specifies the "pres" URL of the PRESENTITY.
- o List of PRESENCE TUPLES
  - Identifier: token to identify this tuple within the presence

information.
- STATUS: OPEN/CLOSED and/or extension status values.
- COMMUNICATION ADDRESS: COMMUNICATION MEANS and CONTACT ADDRESS of this tuple. (optional)
- Relative priority: numerical value specifying the priority of this COMMUNICATION ADDRESS. (optional)
- Timestamp: timestamp of the change of this tuple.(optional)
- Human readable comment: free text memo about this tuple (optional)

o PRESENTITY human readable comment: free text memo about the PRESENTITY (optional).

# 4. XML-encoded Presence Data Format

This section defines an XML-encoded presence information data format (PIDF) for use with CPP compliant systems. A presence payload in this format is expected to be produced by the PRESENTITY (the source of the PRESENCE INFORMATION) and transported to the WATCHERS by the presence servers or gateways without any interpretation or modification.

## 4.1. XML Format Definitions

A PIDF object is a well formed XML document.

It MUST have the XML declaration and it SHOULD contain an encoding declaration in the XML declaration, e.g. "<?xml version='1.0' encoding='UTF-8'?>". If the charset parameter of the MIME content type declaration is present and it is different from the encoding declaration, the charset parameter takes precedence.

Every application conformant to this specification MUST accept the UTF-8 character encoding to ensure the minimal interoperability.

### 4.1.1. The <presence> element

PIDF elements are associated with the XML namespace name 'urn:ietf:params:xml:ns:pidf', declared using an xmlns attribute, per





[XML-NS]. The namespace may be a default namespace, or may be

associated with some namespace prefix (see section 4.2.2 for examples).

The root of an "application/pidf+xml" object is a <presence> element associated with the presence information namespace. This contains any number (including 0) of <tuple> elements, followed by any number (including 0) of <note> elements, followed by any number of OPTIONAL extension elements from other namespaces.

The <presence> element MUST have an 'entity' attribute. The value of the 'entity' attribute is the 'pres' URL of the PRESENTITY publishing this presence document.

The <presence> element MUST contain a namespace declaration ('xmlns') to indicate the namespace on which the presence document is based. The presence document compliant to this specification MUST have the namespace 'urn:ietf:params:xml:ns:pidf:'.

It MAY contain other namespace declarations for the extensions used in the presence XML document.

### 4.1.2. The <tuple> element

The <tuple> element carries a PRESENCE TUPLE, consisting of a mandatory <status> element, followed by any number of OPTIONAL extension elements (possibly from other namespaces), followed by an OPTIONAL <contact> element, followed by any number of OPTIONAL <note> elements, followed by an OPTIONAL <timestamp> element.

Tuples provide a way of segmenting presence information. Protocols or applications may choose to segment the presence information associated with a presentity for any number of reasons - for example, because components of the full presence information for a presentity have come from distinct devices or different applications on the same device, or have been generated at different times. Tuples should be preferred over other manners of segmenting presence information such as creating multiple PIDF instances.

The <tuple> element MUST contain an 'id' attribute which is used to distinguish this tuple from other tuples in the same PRESENTITY. The value of an 'id' attribute MUST be unique within 'id' attribute values of other tuples for the same PRESENTITY. An 'id' value is used by applications processing the presence document to identify the corresponding tuple in the previously acquired PRESENCE INFORMATION of the same PRESENTITY. The value of the 'id' attribute is an arbitrary string, and has no significance beyond providing a means to

distinguish <tuple> values, as noted above.

The <contact> element is OPTIONAL because a PRESENTITY might need to hide its COMMUNICATION ADDRESS or there might be tuples not related to any COMMUNICATION MEANS. Tuples that contain a <basic> status element SHOULD contain a <contact> address. Tuples MAY contain conflicting presence status - one <tuple> might provide a <basic> <status> of OPEN, and another <tuple> in the same PIDF could contain a <basic> <status> of CLOSED, even if they both contain the same <contact> address.

The manner in which segmented presence information is understood by the WATCHER USER AGENT is highly dependent on the capabilities of the WATCHER USER AGENT and the presence application in question. In the absence of any application-specific or protocol-specific understanding of the meaning of tuples, WATCHER USER AGENTS MAY obey the following guidelines. WATCHER USER AGENTS should note which tuples in the PIDF have changed their state since the last notification by correlating the 'id' of each <tuple> with those received in previous notifications and comparing both <status> values and <timestamp> elements in the tuples, if any are present.

### 4.1.3. The <status> element

The <status> element contains one OPTIONAL <basic> element, followed by any number of OPTIONAL extension elements (possibly from other namespaces), under the restriction that at least one child element appears in the <status> element. These children elements of <status> contain status values of this tuple. By allowing multiple status values in a single <tuple> element, different types of status values, e.g. reachability and location, can be represented by a <tuple>. See Section 4.3 for an example with multiple status values.

This memo only defines the <basic> status value element. Other status values may be included using the standard extensibility framework (see Section 4.2.4). Applications encountering unrecognized elements within <status> may ignore them, unless they carry a mustUnderstand="true" or mustUnderstand="1" attribute (see section 4.2.3).

Note that, while the <status> element MUST have at least one status value element, this status value might not be the <basic> element.

### 4.1.4. The <basic> element

The <basic> element contains one of the following strings: "open" or





"closed".

The values "open" and "closed" indicate availability to receive INSTANT MESSAGES if the <tuple> is for an instant messaging address. They also indicate general availability for other communication means, but this memo does not specify these in detail.

open: In the context of INSTANT MESSAGES, this value means that the associated <contact> element, if any, corresponds to an INSTANT INBOX that is ready to accept an INSTANT MESSAGE.

closed: In the context of INSTANT MESSAGES, this value means that the associated <contact> element, if any, corresponds to an INSTANT INBOX that is unable to accept an INSTANT MESSAGE.

### 4.1.5. The <contact> element

The <contact> element contains a URL of the contact address. It optionally has a 'priority' attribute, whose value means a relative priority of this contact address over the others. The value of the attribute MUST be a decimal number between 0 and 1 inclusive with at most 3 digits after the decimal point. Higher values indicate higher priority. Examples of priority values are 0, 0.021, 0.5, 1.00. If the 'priority' attribute is omitted, applications MUST assign the contact address the lowest priority. If the 'priority' value is out of the range, applications just SHOULD ignore the value and process it as if the attribute was not present.

Applications SHOULD handle contacts with a higher priority as they have precedence over those with lower priorities. How they are actually treated is beyond this specification. Also, how to handle contacts with the same priority is up to implementations.

### 4.1.6. The <note> element

The <note> element contains a string value, which is usually used for a human readable comment. A <note> element MAY appear as a child

element of <presence> or as a child element of the <tuple> element. In the former case the comment is about the PRESENTITY and in the latter case the comment is regarding the particular tuple.

Note that, wherever it appears, a <note> element SHOULD NOT be used, and interpreted, as a non-interoperable substitute for status of its parent element.

The <note> element SHOULD have a special attribute 'xml:lang' to





specify the language used in the contents of this element as defined in Section 2.12 of [XML]. The value of this attribute is the language identifier as defined by [RFC3066]. It MAY be omitted when the language used is implied by the larger context such as the encoding information of the contents, such as an xml:lang attribute on an enclosing XML element, or a Content-language header [RFC3282] on an enclosing MIME wrapper.

#### 4.1.7. The <timestamp> element

The <timestamp> element contains a string indicating the date and time of the status change of this tuple. The value of this element MUST follow the IMPP datetime format [RFC3339]. Timestamps that contain 'T' or 'Z' MUST use the capitalized forms.

As a security measure, the <timestamp> element SHOULD be included in all tuples unless the exact time of the status change cannot be determined. For security guidelines for watchers receiving presence information with timestamps, see the Security Considerations.

A PRESENTITY MUST NOT generate successive <presence> elements containing the same timestamp.

### 4.2. Presence Information Extensibility

The presence information extensibility framework is based on XML namespaces [XML-NS].

RFC2779 requires that PIDF have a means of extending <status> values beyond <basic>. These extensions MUST NOT modify how <basic> is to be

understood, nor change the structure or semantics of PIDF bodies themselves. These extensions merely allow protocols and applications to define richer presence data.

### 4.2.1. XML Namespaces Background

All elements and some attributes are associated with a "namespace", which is in turn associated with a globally unique URI. Any developer can introduce their own element names, avoiding conflict by choosing an appropriate namespace URI.

Within the presence data, element or attribute names are associated with a particular namespace by a namespace prefix, which is a leading part of the name, followed by a colon (":"); e.g.





```
<prefix:element-name ...> ... </prefix:element-name>
```

Where, 'prefix' is the header name prefix, 'element-name' is a name which is scoped by the namespace associated with 'prefix'. Note that the choice of 'prefix' is quite arbitrary; it is the corresponding URI that defines the naming scope. Two different prefixes associated with the same namespace URI refer to the same namespace.

A default namespace can be declared for XML elements without a namespace prefix. The default namespace does NOT apply to attribute names, but interpretation of an unprefixed attribute can be determined by the containing element.

A namespace is identified by a URI. In this usage, the URI is used simply as a globally unique identifier, and there is no requirement that it can be used to retrieve a web resource, or for any other purpose. Any legal globally unique URI MAY be used to identify a namespace. (By "globally unique", we mean constructed according to some set of rules so that it is reasonable to expect that nobody else will use the same URI for a different purpose.)

For further details, see the XML namespace specification [XML-NS].

### 4.2.2. XML Namespaces In Presence Information

A URI used as a namespace identifier in PRESENCE INFORMATION data MUST be a full absolute-URI, per RFC 2396 [URI]. (Relative URIs and URI-references containing fragment identifiers MUST NOT be used for this purpose.)

The namespace URI for elements defined by this specification is a URN [URN], using the namespace identifier 'ietf' defined by [URN-NS-IETF] and extended by [XML-Registry]:

```
urn:ietf:params:xml:ns:pidf
```

Thus, simple presence data might be thus:

```
<?xml version="1.0" encoding="UTF-8"?>
<impp:presence xmlns:impp="urn:ietf:params:xml:ns:pidf"
    entity="pres:someone@example.com">
  <impp:tuple id="sg89ae">
    <impp:status>
      <impp:basic>open</impp:basic>
    </impp:status>
    <impp:contact priority="0.8">tel:+09012345678</impp:contact>
  </impp:tuple>
```





```
</impp:presence>
```

or, using a default XML namespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    entity="pres:someone@example.com">
  <tuple id="sg89ae">
    <status>
      <basic>open</basic>
    </status>
    <contact priority="0.8">tel:+09012345678</contact>
  </tuple>
</presence>
```

As is generally the case in XML with namespaces, the xmlns attribute can be used on any element in the presence information to define either the default namespace or a namespace associated with a

namespace prefix.

### 4.2.3. Handling Of Unrecognized Element Names

Except as noted below, a processor of PRESENCE INFORMATION MUST ignore any XML element with an unrecognized name (i.e. having an unrecognized namespace URI, or an unrecognized local name within that namespace). This includes all of the element content, even if it appears to contain elements with recognized names.

Extensions to PIDF are informational in nature - they provide additional information beyond <basic> status. However, in order to understand a complex extension, nested elements within an extension element might need to be marked as mandatory. In such cases, the element name is qualified with a mustUnderstand='true' or mustUnderstand='1' attribute. See section 4.3.3 for an example.

> NOTE: a mustUnderstand='true' or mustUnderstand='1' attribute within an element that is being ignored is itself ignored. The writer of nested mandatory-to-understand information is responsible for ensuring that any enclosing element is also labelled with a mustUnderstand='true' or mustUnderstand='1' attribute, if necessary.

This specification defines (section 4.1) elements within the 'urn:ietf:params:xml:ns:pidf' namespace that MUST be recognized in CPP presence data. Processors MUST handle these as described, even if they do not carry a mustUnderstand attribute. The XML Schema Definition (section 4.4) indicates those elements that MUST be





present in a valid presence information document.

If an agent receives PRESENCE INFORMATION with a <status> block containing an unrecognized element with a mustUnderstand='true' (or '1') attribute, it should treat that entire element and any content as unrecognized and not attempt to process it.

In order to ensure that minimal implementations can correctly process basic PIDF information the mustUnderstand attribute MUST be used only within optional elements nested in a <status> element. This will ensure that problems processing an extension are restricted to that

extension and do not affect the processing of the basic PIDF information defined in this specification.

### 4.2.4. Status Value Extensibility

This memo defines only the <basic> status value with values of "open" and "closed". Other status values are possible using the standard namespace-based extensibility rules defined above.

For example, a location status value might be included thus:

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:local="urn:example-com:pidf-status-type"
    entity="pres:someone@example.com">
  <tuple id="ub93s3">
    <status>
      <basic>open</basic>
      <local:location>home</local:location>
    </status>
    <contact>im:someone@example.com</contact>
  </tuple>
</presence>
```

Some new status values will 'extend' the value of the <basic> element. For example, a status value defined for use with instant messaging may include values such as 'away', 'busy' and 'offline'. In order that some level of interoperability be maintained with user agents that don't recognize the new extension, the <basic> status value must also be included. This means that extensions are not obligated to define a mapping from each of their values to OPEN or CLOSED.

### 4.2.5. Standardizing Status Extensions





Although the existing PIDF definition allows arbitrary elements to appear in the <status> element, it may be sometimes desirable to standardize extension status elements and their semantics (the meanings of particular statuses, how they should be interpreted). The

URN 'urn:ietf:params:xml:ns:pidf:status' has been defined as a namespace URI for extensions standardized by the IETF, and new values in this namespace must be defined by a standards-track RFC.

The following example XML Schema defines an extension for <location> presence information, which can have the values of 'home', 'office', or 'car'. If the <location> element were standardized, this document would be made available in an RFC along with information about the use of the extension. These extensions should use the namespace 'urn:ietf:params:xml:ns:pidf:status', and each RFC defining an extension should register an extension name within that namespace with IANA.

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

  <xs:simpleType name="location">
    <xs:restriction base="xs:string">
      <xs:enumeration value="home"/>
      <xs:enumeration value="office"/>
      <xs:enumeration value="car"/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

In addition to the XML Schema to validate the extension, registration of the extension name with IANA, RFCs defining extensions MUST discuss:

- The domain of applicability of the extension. Is this extension exclusively valuable to IM clients, telephones, geolocators, etc? What sorts of presence applications would use this extension and under what circumstances?

- Semantics for the presence states defined in the extension. What disposition provokes an automated presentity to declare that it is in state X, or does a human select X from a drag-down menu? Is there any general guidance for watchers of presence information with state Y (for example, how they should best attempt to

communicate with the presentity, if at all, when the principal is in state Y).

Extensions SHOULD also discuss:

- How, if at all, any presence states defined in the extension related to <basic>, or to any relevant extension previously published in an RFC. For example, "state Z implies OPEN, so it MUST NOT be used if a basic state of CLOSED is expressed", or "you should use the extension in this document, not the extension in RFC QQQQ, if your circumstances are as follows...."

## 4.3. Examples

### 4.3.1. Default Namespace with Status Extensions

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
     xmlns:im="urn:ietf:params:xml:ns:pidf:im"
     xmlns:myex="http://id.example.com/presence/"
     entity="pres:someone@example.com">
  <tuple id="bs35r9">
    <status>
      <basic>open</basic>
      <im:im>busy</im:im>
      <myex:location>home</myex:location>
    </status>
    <contact priority="0.8">im:someone@mobilecarrier.net</contact>
    <note xml:lang="en">Don't Disturb Please!</note>
    <note xml:lang="fr">Ne derangez pas, s'il vous plait</note>
    <timestamp>2001-10-27T16:49:29Z</timestamp>
  </tuple>
  <tuple id="eg92n8">
    <status>
      <basic>open</basic>
    </status>
    <contact priority="1.0">mailto:someone@example.com</contact>
  </tuple>
  <note>I'll be in Tokyo next week</note>
</presence>
```

### 4.3.2. Presence with Other Extension Elements

```
<?xml version="1.0" encoding="UTF-8"?>
<impp:presence xmlns:impp="urn:ietf:params:xml:ns:pidf"
```

```
        xmlns:myex="http://id.example.com/presence/"
```





```
        entity="pres:someone@example.com">
    <impp:tuple id="ck38g9">
      <impp:status>
        <impp:basic>open</impp:basic>
      </impp:status>
      <myex:mytupletag>Extended value in tuple</myex:mytupletag>
      <impp:contact priority="0.65">tel:+09012345678</impp:contact>
    </impp:tuple>
    <impp:tuple id="md66je">
      <impp:status>
        <impp:basic>open</impp:basic>
      </impp:status>
      <impp:contact priority="1.0">
         im:someone@mobilecarrier.net</impp:contact>
    </impp:tuple>
    <myex:mytag>My extended presentity information</myex:mytag>
  </impp:presence>
```

### 4.3.3. Example Mandatory To Understand Elements

```
  <?xml version="1.0" encoding="UTF-8"?>
  <impp:presence xmlns:impp="urn:ietf:params:xml:ns:pidf"
       xmlns:myex="http://id.mycompany.com/presence/"
       entity="pres:someone@example.com">
    <impp:tuple id="tj25ds">
      <impp:status>
        <impp:basic>open</impp:basic>
      </impp:status>
      <myex:complexExtension>
        <myex:ex1 impp:mustUnderstand="1">val1</myex:ex1>
        <myex:ex2>val2</myex:ex2>
      </myex:complexExtension>
      <impp:contact priority="0.725">tel:+09012345678</impp:contact>
    </impp:tuple>
    <myex:mytag>My extended presentity information</myex:mytag>
  </impp:presence>
```

Here, <myex:ex1> must be understood and, if it is not recognized, <myex:complexExtension> MUST be ignored. <myex:mytag> and

<myex:ex2> MAY be ignored if they are not recognized.

## 4.4. XML Schema Definitions

This section gives the XML Schema Definition [XMLSchema1] of the "application/pidf+xml" format. This is presented as a formal definition of the "application/pidf+xml" format. Note that the XML





Schema definition is not intended to be used with on-the-fly validation of the presence XML document.

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

  <xs:element name="presence" type="tns:presence"/>

  <xs:complexType name="presence">
    <xs:sequence>
      <xs:element name="tuple" type="tns:tuple" minOccurs="0"
         maxOccurs="unbounded"/>
      <xs:element name="note" type="tns:note" minOccurs="0"
         maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="entity" type="xs:anyURI" use="required"/>
  </xs:complexType>

  <xs:complexType name="tuple">
    <xs:sequence>
      <xs:element name="status" type="tns:status"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded"/>
```

```
        <xs:element name="contact" type="tns:contact" minOccurs="0"/>
        <xs:element name="note" type="tns:note" minOccurs="0"
           maxOccurs="unbounded"/>
        <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
    </xs:complexType>

    <xs:complexType name="status">
      <xs:sequence>
        <xs:element name="basic" type="tns:basic" minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
           maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
```





```
    <xs:simpleType name="basic">
      <xs:restriction base="xs:string">
        <xs:enumeration value="open"/>
        <xs:enumeration value="closed"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="contact">
      <xs:simpleContent>
        <xs:extension base="xs:anyURI">
          <xs:attribute name="priority" type="tns:qvalue"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

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

    <xs:simpleType name="qvalue">
      <xs:restriction base="xs:decimal">
        <xs:pattern value="0(.[0-9]{0,3})?"/>
```

```
        <xs:pattern value="1(.0{0,3})?"/>
      </xs:restriction>
    </xs:simpleType>

    <!-- Global Attributes -->
    <xs:attribute name="mustUnderstand" type="xs:boolean" default="0">
      <xs:annotation>
        <xs:documentation>
        This attribute may be used on any element within an optional
        PIDF extension to indicate that the corresponding element must
        be understood by the PIDF processor if the enclosing optional
        element is to be handled.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:schema>
```

## 5. IANA Considerations

This memo calls for IANA to:
- register a new MIME content-type application/pidf+xml,





per [RFC 2048],
- register a new XML namespace URN per [XML-Registry].
- register a new XML namespace URN for status extensions per [XML-Registry].

The registration templates for these are below. For more information on status extensions, see section 4.2.5.

### 5.1. Content-type registration for 'application/pidf+xml'

To: ietf-types@iana.org
Subject: Registration of MIME media type application/pidf+xml

MIME media type name: application

MIME subtype name: pidf+xml

Required parameters: (none)

Optional parameters: charset
Indicates the character encoding of enclosed XML. Default is UTF-8.

Encoding considerations:
Uses XML, which can employ 8-bit characters, depending on the character encoding used.
See RFC 3023 [RFC 3023], section 3.2.

Security considerations:
This content type is designed to carry presence data, which may be considered private information. Appropriate precautions should be adopted to limit disclosure of this information.

Interoperability considerations:
This content type provides a common format for exchange of presence information across different CPP compliant protocols.

Published specification:
RFCXXXX (this document)

Applications which use this media type:
Presence and instant messaging systems.

Additional information:

Magic number(s):
File extension(s):





Macintosh File Type Code(s):

Person & email address to contact for further information:
Hiroyasu Sugano
E-mail: sugano.h@jp.fujitsu.com

Intended usage:
LIMITED USE

Author/Change controller:
This specification is a work item of the IETF IMPP working group, with mailing list address <impp@iastate.edu>.

Other information:
This media type is a specialization of application/xml [RFC 3023], and many of the considerations described there also apply to application/pidf+xml.

## 5.2. URN sub-namespace registration for 'urn:ietf:params:xml:ns:pidf'

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

Description:
This is the XML namespace URI for XML elements defined by [RFCXXXX] to describe CPP presence information in application/pidf+xml content type.

Registrant Contact
IETF, IMPP working group, <impp@iastate.edu>
Hiroyasu Sugano, <sugano.h@jp.fujitsu.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=utf-8"/>
  <title>Namespace for CPP presence information</title>
</head>
<body>
  <h1>Namespace for CPP presence information</h1>
  <h2>application/pidf+xml</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
```





```
  </body>
  </html>
END
```

## 5.3. URN sub-namespace registration for 'urn:ietf:params:xml:ns:pidf:status'

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

Description:
This is the XML namespace URI for XML elements defined by [RFCXXXX] to describe extensions to the status of CPP presence information in application/pidf+xml content type.

Registrant Contact
IETF, IMPP working group, <impp@iastate.edu>
Hiroyasu Sugano, <sugano.h@jp.fujitsu.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=utf-8"/>
  <title>Namespace for CPP status extensions</title>
</head>
<body>
  <h1>Namespace for CPP presence information extensions</h1>
  <h2>application/pidf+xml</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
```

END

# 6. Security Considerations

Because presence is very privacy-sensitive information, the protocol for the presence information MUST have capabilities to protect PIDF from possible threats, such as eavesdropping, corruption, tamper and replay attacks. These security mechanisms must be able to be used end-to-end between presentities and watchers, even if the watcher and the presentity employ different presence protocols and communicate

through a CPP gateway. Since the 'application/pidf+xml' MIME type is defined for this PIDF document, staging security for PIDF at the MIME level (with S/MIME [RFC2633]) seems appropriate. Therefore, PIDF should follow the normative recommendations for the use of S/MIME (including minimum ciphersuites) given in the core CPP specification.

Note that the use of timestamps in PIDF (see section 4.1.7) can provide some rudimentary protection against replay attacks. If a watcher receives presence information that is outdated, it SHOULD be ignored. A watcher can determine that presence information is outdated in a number of fashions. Most significantly, if the newest timestamp in presence information is older than the newest timestamp in the last received presence information, it should be considered outdated. Applications and protocols also are advised to adopt their own rules for determining how frequently presence information should be refreshed. For example, if presence information appears to be more than one hour old, it could be considered outdated (a notification generated for this presence information will not take such a long time to reach a watcher, and if a presentity has not refreshed its presence state in the last hour, it is probably offline).

## 7. Internationalization Considerations

All the processors conformant to this specification MUST be able to generate and accept UTF-8 encoding, this being one of the mandatory character encodings for XML conforming processors, and also required by the policies set out in RFC 2277 [RFC2277].

Other character encodings MAY be accepted (but CPP compliant processors are strongly discouraged from emitting anything other than UTF-8).

## 8. Normative References

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

[RFC3023] M. Murata, S. St.Laurent, D. Kohn, "XML Media Types", RFC 3023, January 2001.

[XML] T. Bray, J. Paoli, C. Sperberg-McQueen and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, October 2000, <http://www.w3.org/TR/2000/REC-xml-20001006>

[MIME] Multipurpose Internet Mail Extensions. See RFC 2045,

RFC 2046, RFC 2047, RFC 2048, and RFC 2049.

[RFC3066] H. Alvestrand, "Tags for the Identification of Languages", RFC 3066, March 1995.

[RFC3339] G. Klyne and C.Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[XML-NS] Tim Bray, Dave Hollander, and Andrew Layman "Namespaces in XML", W3C recommendation: xml-names, 14 January 1999, <http://www.w3.org/TR/REC-xml-names>

[URI] T. Berners-Lee, R.T.Fielding and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

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

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

[XML-Registry] M. Mealling, "The IETF XML Registry", draft-mealling-iana-xmlns-registry-04, Work in Progress.

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

[XMLSchema1] H. Thompson, D. Beech, M. Maloney and N. Mendelsohn, "XML Schema Part 1: Structures", W3C REC-xmlschema-1, May 2001, <http://www.w3.org/TR/xmlschema-1/>.

## 9. Informative References

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

[RFC2779] M. Day, S. Aggarwal, G. Mohr, and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.

[CPIM] D. Crocker and J. Peterson, "Common Profile for Instant Messaging (CPIM)", draft-ietf-impp-im-02.txt, Work in Progress.

[CPP] D. Crocker and J. Peterson, "Common Presence for Presence (CPP)", draft-ietf-impp-pres-02.txt, Work in Progress.

[CPIM-MSG] D. Atkins and G. Klyne, "Common Presence and Instant Messaging Message Format", draft-ietf-impp-cpim-msgfmt-08.txt, Work in Progress.





[vCard] F. Dawson and T. Howes, "vCard MIME Directory Profile", RFC 2426, September 1998.

[RFC2633] B. Ramsdell, "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[RFC3282] H. Alvestrand, "Content Language Headers", RFC 3282, May 2002.

## 10. Authors' Addresses

Hiroyasu Sugano
Fujitsu Laboratories Ltd.
64, Nishiwaki
Ohkubo-cho
Akashi 674-8555
Japan
E-mail: sugano.h@jp.fujitsu.com

Shingo Fujimoto
Fujitsu Laboratories Ltd.
64, Nishiwaki
Ohkubo-cho
Akashi 674-8555
Japan
E-mail: shingo_fujimoto@jp.fujitsu.com

Graham Klyne
Nine by Nine
E-mail: GK@ninebynine.org

Adrian Bateman
VisionTech Limited
Colton, Staffordshire, WS15 3LD

United Kingdom
E-mail: bateman@acm.org

Wayne Carr
Intel Corporation
2111 NE 25th Avenue
Hillsboro, OR 97124
USA
E-mail: wayne.carr@intel.com

Jon Peterson
NeuStar, Inc.
1800 Sutter St





Suite 570
Concord, CA 94520
USA
Phone: +1 925/363-8720
E-mail: jon.peterson@neustar.biz

## 11. Appendix A. Document Type Definitions

The Document Type Definition for the "application/pidf+xml" format is described. The DTD here is presented only for informational for those who may not familiar with the XML Schema definition.

Note: the DTD does not show where extension elements can be added. See the XML Schema for that information.

```
<!ENTITY % URL          "CDATA">
<!ENTITY % URI          "CDATA">
<!ENTITY % TUPLEID      "CDATA">
<!ENTITY % DATETIME     "CDATA">
<!ENTITY % VALUETYPE    "CDATA">
<!ENTITY % PRIORITY     "CDATA">
<!ENTITY % NOTE         "CDATA">

<!ELEMENT presence ((tuple*),note?)>
<!ATTLIST presence
          xmlns     %URI;     #REQUIRED
```

```
          entity    %URL;     #REQUIRED
>

<!ELEMENT tuple (status,contact?,note?,timestamp?)>
<!ATTLIST tuple
          id   %TUPLEID;      #REQUIRED
>

<!ELEMENT status (basic?)>
<!ELEMENT basic CDATA>

<!ELEMENT contact %URL;>
<!ATTLIST contact
          priority %PRIORITY; #IMPLIED
>

<!ELEMENT note %NOTE;>

<!ELEMENT timestamp %DATETIME;>
```





## 12. Full Copyright Statement