Internet Engineering Task Force (IETF)
Request for Comments: 8048
Obsoletes: 7248
Category: Standards Track
ISSN: 2070-1721

P. Saint-Andre
Filament
December 2016

Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Presence

Abstract

This document defines a bidirectional protocol mapping for the exchange of presence information between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP). This document obsoletes RFC 7248.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc8048.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

```
   1.  Introduction  . . . . . . . . . . . . . . . . . . . . . . . .   3
   2.  Intended Audience . . . . . . . . . . . . . . . . . . . . . .   4
   3.  Terminology . . . . . . . . . . . . . . . . . . . . . . . . .   5
   4.  Architectural Assumptions . . . . . . . . . . . . . . . . . .   5
   5.  Presence Authorizations . . . . . . . . . . . . . . . . . . .   6
     5.1.  Overview  . . . . . . . . . . . . . . . . . . . . . . . .   6
     5.2.  XMPP to SIP . . . . . . . . . . . . . . . . . . . . . . .   7
       5.2.1.  Requesting a Presence Authorization . . . . . . . . .   7
       5.2.2.  Refreshing a Notification Dialog  . . . . . . . . . .  11
       5.2.3.  Cancelling a Presence Authorization . . . . . . . . .  11
     5.3.  SIP to XMPP . . . . . . . . . . . . . . . . . . . . . . .  15
       5.3.1.  Requesting a Presence Authorization . . . . . . . . .  15
       5.3.2.  Refreshing a Notification Dialog  . . . . . . . . . .  18
       5.3.3.  Cancelling a Presence Authorization . . . . . . . . .  19
   6.  Notifications of Presence Information . . . . . . . . . . . .  19
     6.1.  Overview  . . . . . . . . . . . . . . . . . . . . . . . .  19
     6.2.  XMPP to SIP . . . . . . . . . . . . . . . . . . . . . . .  20
     6.3.  SIP to XMPP . . . . . . . . . . . . . . . . . . . . . . .  25
   7.  Polling for Presence Information  . . . . . . . . . . . . . .  27
     7.1.  XMPP to SIP . . . . . . . . . . . . . . . . . . . . . . .  27
     7.2.  SIP to XMPP . . . . . . . . . . . . . . . . . . . . . . .  28
   8.  Privacy and Security Considerations . . . . . . . . . . . . .  28
     8.1.  Amplification Attacks . . . . . . . . . . . . . . . . . .  28
     8.2.  Presence Leaks  . . . . . . . . . . . . . . . . . . . . .  29
   9.  References  . . . . . . . . . . . . . . . . . . . . . . . . .  30
     9.1.  Normative References  . . . . . . . . . . . . . . . . . .  30
     9.2.  Informative References  . . . . . . . . . . . . . . . . .  31
   Appendix A.  Changes from RFC 7248  . . . . . . . . . . . . . . .  33
   Acknowledgements  . . . . . . . . . . . . . . . . . . . . . . . .  34
   Author's Address  . . . . . . . . . . . . . . . . . . . . . . . .  34
```

1. Introduction

Presence is information about the availability of an entity (such as network availability or availability for communication). Presence features in both SIP and XMPP involve several aspects:

- A long-lived authorization for a user to receive notifications about a contact’s presence across presence and notification sessions; such an authorization is formally requested by the user, approved (or not) by the contact, and often associated with a record in an address list or "buddy list".

- An ephemeral presence session, during which the contact is online (i.e., available for interaction) and after which the contact is offline again.

- An ephemeral notification session, during which the user requests presence notifications from the contact (these are implicit in XMPP, but explicit in SIP where they are managed by means of notification dialogs).

- Notifications that are sent from the contact to the user for the life of either the contact’s presence session or the user’s notification session.

Although specifications for both SIP and XMPP use the term "subscription", they do so in different ways. In SIP, a "subscription" is the specific mechanism whereby a subscriber (or an entity acting on the subscriber’s behalf, such as a SIP Presence Server) requests presence notifications from the contact over a relatively short period of time, renewed as necessary to keep receiving presence notifications during a presence session. By contrast, in XMPP a "subscription" is essentially shorthand for a long-lived presence authorization. To prevent confusion, this document uses the term "notification dialog" for a SIP subscription and the term "presence authorization" for an XMPP subscription.

In order to help ensure interworking between presence systems that conform to the instant messaging and presence protocol requirements [RFC2779], it is important to clearly define protocol mappings between such systems. Within the IETF, work has proceeded on two presence technologies:

- Various extensions to the Session Initiation Protocol ([RFC3261]) for presence, in particular [RFC3856]

- The Extensible Messaging and Presence Protocol (XMPP), which consists of a formalization of the core XML-streaming protocols developed originally by the Jabber open-source community; the relevant specifications are [RFC6120] for the XML-streaming layer and [RFC6121] for basic presence and instant-messaging extensions

One approach to help ensure interworking between these protocols is to map each protocol to the abstract semantics described in [RFC3860]; however, apparently that approach has never been implemented. The approach taken in this document is to directly map semantics from one protocol to another (i.e., from SIP/SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) to XMPP and vice versa), because that is how existing systems solve the interworking problem.

The architectural assumptions underlying such direct mappings are provided in [RFC7247], including mapping of addresses and error conditions. The mappings specified in this document cover basic presence functionality. Mapping of more advanced functionality (e.g., so-called "rich presence") is out of scope for this document.

This document obsoletes RFC 7248.

2. Intended Audience

The documents in this series (which include [RFC7247], [RFC7572], [RFC7573], and [RFC7702]) are intended for use by software developers who have an existing system based on one of these technologies (e.g., SIP) and would like to enable communication from that existing system to systems based on the other technology (e.g., XMPP). We assume that readers are familiar with the core specifications for both SIP [RFC3261] and XMPP [RFC6120], with the base document for this series [RFC7247], and with the following presence-related specifications:

- "A Presence Event Package for the Session Initiation Protocol" [RFC3856]

- "Presence Information Data Format (PIDF)" [RFC3863]

- "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence" [RFC6121]

- "SIP-Specific Event Notification" [RFC6665]

3. Terminology

A number of terms used here ("user", "contact", "notification", etc.) are explained in [RFC3261], [RFC3856], [RFC3857], [RFC6120], and [RFC6121]. This document uses some, but not all, of the presence-related terms defined in the Model for Presence and Instant Messaging [RFC2778]. In particular, the term "presence session" is used as described in [RFC6121] to mean a delimited time period during which an endpoint is online and available for communications.

In flow diagrams, SIP traffic is shown using arrows such as "***>", whereas XMPP traffic is shown using arrows such as "...>". As in [RFC7247], the terms "SIP to XMPP Gateway" and "XMPP to SIP Gateway" are abbreviated as "S2X GW" and "X2S GW", respectively.

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

4. Architectural Assumptions

The fundamental architectural assumptions underlying SIP-XMPP interworking are described in [RFC7247].

Note that, in SIP, there are two ways that presence services can be deployed on the server side:

1. Under this model, described most fully in [RFC3857], a dedicated SIP Presence Server handles events related to the presence event package. Instead of forwarding a SUBSCRIBE message to the SIP user, the Presence Server would inform the user of subscription activity using the ’presence.winfo’ event package. The SIP User Agent would then authorize the subscribing contact through some interaction with the Presence Server (for instance, using XML Configuration Access Protocol (XCAP) [RFC4825]). Therefore, presence updates from the SIP User Agent would not be sent as NOTIFY messages to the XMPP user but as PUBLISH messages to the Presence Server, which would then generate NOTIFY messages to all active subscribers.

2. Under this model, a SIP Presence Server acts in proxy mode and merely passes through the SUBSCRIBE and NOTIFY messages to the SIP User Agent.

Because the behavior of the XMPP-to-SIP gateway is not changed by the SIP architectural model that is used, the diagrams and protocol flows in this document cover both options by labeling the end entity a "SIP User Agent or Presence Server".

5. Presence Authorizations

5.1. Overview

Both XMPP and presence-aware SIP systems enable entities (often, but not necessarily, human users) to subscribe to the presence of other entities. XMPP presence is specified in [RFC6121]. Presence using a SIP event package is specified in [RFC3856].

As described in [RFC6121], XMPP presence authorizations are managed using XMPP <presence/> stanzas of type "subscribe", "subscribed", "unsubscribe", and "unsubscribed". The main states are:

o "none" (neither the user nor the contact is subscribed to the other’s presence information)

o "from" (the contact will receive presence notifications from the user)

o "to" (the contact will send presence notifications to the user)

o "both" (both user and contact will receive each other’s presence notifications)

As described in [RFC3856], in SIP the subscriber does not explicitly request the creation or removal of presence authorizations. Rather, the authorizations are triggered by subscription activity. When a SIP user receives an initial SIP SUBSCRIBE event from a contact, the recipient’s SIP User Agent or SIP Presence Server asks the user to make an authorization policy decision. This decision is recorded in the SIP User Agent or SIP Presence Server, so that in the future any notification dialogs from the contact are automatically approved. (Note that addresses for SIP users and contacts are most generally referenced by a Presence URI of the form <pres:user@domain> but might be referenced by a SIP or SIPS (Session Initiation Protocol Secure) URI of the form <sip:user@domain> or <sips:user@domain>; because, in practice, ’pres’ URIs are rarely used, the examples in this document use ’sip’ URIs.)

In both SIP and XMPP, presence authorizations are long-lived (indeed permanent if not explicitly cancelled). In SIP, by default a notification session is typically short-lived unless explicitly extended (the default time-to-live of a SIP notification dialog is

3600 seconds, as specified in Section 6.4 of [RFC3856], so that a notification dialog needs to be explicitly refreshed in order for a user’s notification session to last as long as the contact’s presence session). In XMPP, a user’s notification session with a contact is almost always automatically handled by the user’s server based on the user’s presence state (see [RFC6121] for details).

5.2. XMPP to SIP

5.2.1. Requesting a Presence Authorization

The following diagram illustrates the protocol flow necessary to establish an authorization for an XMPP user to a receive presence notifications from a SIP contact, as further explained in the text and examples after the diagram.

```
   XMPP         XMPP         SIP     SIP UA or
   Client      Server       Proxy    Presence Server
    |        + X2S GW         |            |
    |            |            |            |
    | (F1) XMPP  |            |            |
    | subscribe  |            |            |
    |...........>|            |            |
    |            | (F2) SIP   |            |
    |            | SUBSCRIBE  |            |
    |            |***********>|            |
    |            |            | (F3) SIP   |
    |            |            | SUBSCRIBE  |
    |            |            |***********>|
    |            |            | (F4) SIP   |
    |            |            | 200 OK     |
    |            |            |<***********|
    |            | (F5) SIP   |            |
    |            | 200 OK     |            |
    |            |<***********|            |
    |            |            | (F6) SIP   |
    |            |            | NOTIFY     |
    |            |            | (pending)  |
    |            |            |<***********|
    |            | (F7) SIP   |            |
    |            | NOTIFY     |            |
    |            |<***********|            |
    |            | (F8) SIP   |            |
    |            | 200 OK     |            |
    |            |***********>|            |
    |            |            | (F9) SIP   |
    |            |            | 200 OK     |
    |            |            |***********>|
    |            |            |            |
    |            |            | (F10) SIP  |
    |            |            | NOTIFY     |
    |            |            | (active)   |
    |            |            |<***********|
    |            | (F11) SIP  |            |
    |            | NOTIFY     |            |
    |            |<***********|            |
    |            | (F12) SIP  |            |
    |            | 200 OK     |            |
    |            |***********>|            |
    |            |            | (F13) SIP  |
    |            |            | 200 OK     |
    |            |            |***********>|
    | (F14) XMPP |            |            |
    | subscribed |            |            |
    |<...........|            |            |
    | (F15) XMPP |            |            |
    | presence   |            |            |
    |<...........|            |            |
    |            |            |            |
```

An XMPP user (e.g., juliet@example.com) asks for a presence authorization by sending a request to a SIP contact (e.g., romeo@example.net), and the contact either accepts or declines the request. If the SIP contact accepts the request, the XMPP user will have a long-lived authorization to receive the SIP contact’s presence information until (1) the XMPP user unsubscribes or (2) the SIP contact cancels the authorization. The request is encapsulated in a <presence/> stanza of type "subscribe":

Example 1: XMPP User Subscribes to SIP Contact (F1)

```
|  <presence from='juliet@example.com'
|            to='romeo@example.net'
|            type='subscribe'/>
```

Upon receiving such a <presence/> stanza, the XMPP server to which Juliet has connected needs to determine the identity of the domainpart in the ’to’ address, which it does by following the procedures explained in Section 5 of [RFC7247]. If the domain is a SIP domain, the XMPP server will hand off the <presence/> stanza to an associated XMPP-to-SIP gateway or connection manager that natively communicates with presence-aware SIP proxies.

The XMPP-to-SIP gateway is then responsible for translating the XMPP request into a SIP SUBSCRIBE request addressed from the XMPP user to the SIP contact:

Example 2: SIP Transformation of XMPP Presence Authorization Request (F2)

```
|  SUBSCRIBE sip:romeo@example.net SIP/2.0
|  Via: SIP/2.0/TCP x2s.example.com;branch=z9hG4bKna998sk
|  From: <sip:juliet@example.com>;tag=j89d
|  Call-ID: 5BCF940D-793D-43F8-8972-218F7F4EAA8C
|  Event: presence
|  Max-Forwards: 70
|  CSeq: 1 SUBSCRIBE
|  Contact: <sip:juliet@example.com>;gr=yn0cl4bnw0yr3vym
|  Accept: application/pidf+xml
|  Expires: 3600
|  Content-Length: 0
```

Once the SIP proxy has delivered the SIP SUBSCRIBE to the SIP User Agent or Presence Server (F3, no example shown), the SIP User Agent would then send a response indicating acceptance of the request:

Example 3: SIP User Accepts Presence Authorization Request (F4)

```
|  SIP/2.0 200 OK
|  Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
|  From: <sip:juliet@example.com>;tag=j89d
|  To: <sip:romeo@example.net>;tag=ffd2
|  Call-ID: 5BCF940D-793D-43F8-8972-218F7F4EAA8C
|  CSeq: 1 SUBSCRIBE
|  Contact: <sip:romeo@example.net>;gr=dr4hcr0st3lup4c
|  Expires: 3600
|  Content-Length: 0
```

In accordance with Section 6.7 of [RFC3856], the XMPP-to-SIP gateway needs to consider the state to be "neutral" until it receives a NOTIFY message with a Subscription-State header [RFC6665] whose value is "active". Therefore, the SIP User Agent or Presence Server SHOULD immediately send such a NOTIFY message (see Section 6 below). If the XMPP-to-SIP gateway initially receives one or more NOTIFY messages with a Subscription-State header whose value is "pending" (F6), then it MUST respond to them on the SIP side but refrain from sending any presence stanzas from the SIP contact to the XMPP user.

Example 4: SIP User Agent or Presence Server Sends Presence Notification (F10)

```
|  NOTIFY sip:juliet@example.com SIP/2.0
|  Via: SIP/2.0/TCP simple.example.net;branch=z9hG4bKna998sk
|  From: <sip:juliet@example.com>;tag=j89d
|  To: <sip:romeo@example.net>;tag=ffd2
|  Call-ID: 5BCF940D-793D-43F8-8972-218F7F4EAA8C
|  Event: presence
|  Subscription-State: active;expires=499
|  Max-Forwards: 70
|  CSeq: 2 NOTIFY
|  Content-Type: application/pidf+xml
|  Content-Length: 193
|
|  <?xml version='1.0' encoding='UTF-8'?>
|  <presence xmlns='urn:ietf:params:xml:ns:pidf'
|            entity='pres:romeo@example.net'>
|    <tuple id='ID-dr4hcr0st3lup4c'>
|      <status>
|        <basic>open</basic>
|        <show xmlns='jabber:client'>away</show>
|      </status>
|    </tuple>
|  </presence>
```

Upon receiving the first NOTIFY with a state of active, the XMPP-to-SIP gateway returns a 200 OK to the SIP User Agent or Presence Server (F12, no example shown).

The XMPP-to-SIP gateway also generates a <presence/> stanza of type "subscribed":

Example 5: XMPP User Receives Acknowledgement from SIP Contact (F14)

```
|  <presence from='romeo@example.net'
|            to='juliet@example.com'
|            type='subscribed'/>
```

As described in Section 6, if this first NOTIFY in the notification session contains a body, then the XMPP-to-SIP gateway also generates a presence notification addressed to the XMPP user (if the NOTIFY does not contain a body, then the gateway would interpret it as unknown or "closed"):

Example 6: XMPP User Receives Presence Notification from SIP Contact (F15)

```
|  <presence from='romeo@example.net/dr4hcr0st3lup4c'
|            to='juliet@example.com'/>
```

5.2.2. Refreshing a Notification Dialog

It is the responsibility of the XMPP-to-SIP gateway to set the value of the Expires header and to periodically renew the notification dialog on the SIMPLE side of the gateway. For example, the XMPP-to-SIP gateway SHOULD send a new SUBSCRIBE request to the SIP contact whenever the XMPP user initiates a presence session with the XMPP server by sending initial presence to its XMPP server (this is functionally equivalent to sending an XMPP presence probe). The XMPP-to-SIP gateway SHOULD also send a new SUBSCRIBE request to the SIP contact sufficiently in advance of when the SIP notification dialog is scheduled to expire during the XMPP user's active presence session.

The rules regarding SIP SUBSCRIBE requests for the purpose of establishing and refreshing a notification dialog are provided in [RFC6665]. Those rules also apply to XMPP-to-SIP gateways. Furthermore, an XMPP-to-SIP gateway MUST consider the XMPP presence authorization to be permanently cancelled (and so inform the XMPP user) if it receives a SIP response of 403, 489, or 603. By contrast, it is appropriate to consider a SIP response of 423 or 481 to be a transient error and to honor the long-lived XMPP presence authorization. [RFC6665] explains more detailed considerations about the handling of SIP responses in relation to notification dialogs and refreshes.

Finally, see the Privacy and Security Considerations section (Section 8) for important information and requirements regarding the security implications of notification refreshes.

5.2.3. Cancelling a Presence Authorization

The following diagram illustrates the protocol flow by which an XMPP user cancels her outbound presence authorization with a SIP contact (i.e., indicates that she no longer wishes to be authorized to see the SIP contact's presence). As can be seen, SIMPLE itself does not have a construct that enables a user to cancel her outbound presence authorization (however, in many SIP/SIMPLE implementations she could use a technology such as XCAP [RFC4825] to remove the contact from her address list); therefore, this flow instead results in the cancellation of the user’s notification dialog (with the implication on the XMPP side that the user will not request a subsequent notification dialog). Additional details are explained in the text and examples after the diagram.

```
   XMPP         XMPP         SIP        SIP UA or
   Client      Server       Proxy    Presence Server
    |        + X2S GW         |            |
    |            |            |            |
    | (F16) XMPP |            |            |
    |unsubscribe |            |            |
    |...........>|            |            |
    |            | (F17) SIP  |            |
    |            | SUBSCRIBE  |            |
    |            | Expires: 0 |            |
    |            |***********>|            |
    |            |            | (F18) SIP  |
    |            |            | SUBSCRIBE  |
    |            |            | Expires: 0 |
    |            |            |***********>|
    |            |            | (F19) SIP  |
    |            |            | 200 OK     |
    |            |            |<***********|
    |            | (F20) SIP  |            |
    |            | 200 OK     |            |
    |            |<***********|            |
    | (F21) XMPP |            |            |
    |unsubscribed|            |            |
    |<...........|            |            |
    |            | (F22) SIP  |            |
    |            | NOTIFY     |            |
    |            | terminated |            |
    |            |***********>|            |
    |            |            | (F23) SIP  |
    |            |            | NOTIFY     |
    |            |            | terminated |
    |            |            |***********>|
    |            |            | (F24) SIP  |
    |            |            | 200 OK     |
    |            |            |<***********|
    |            | (F25) SIP  |            |
    |            | 200 OK     |            |
    |            |<***********|            |
    |            |            |            |
```

At any time after subscribing, the XMPP user can indicate that she no longer wishes to be authorized to receive presence notifications from the contact. This is done by sending a <presence/> stanza of type "unsubscribe":

Example 7: XMPP User Unsubscribes from SIP Contact (F16)

```
|  <presence from='juliet@example.com'
|            to='romeo@example.net'
|            type='unsubscribe'/>
```

The XMPP-to-SIP gateway is responsible for translating the XMPP unsubscribe command into a SIP SUBSCRIBE request with the Expires header set to a value of zero ("0"):

Example 8: SIP Transformation of XMPP Unsubscribe (F17)

```
|  SUBSCRIBE sip:romeo@example.net SIP/2.0
|  Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
|  From: <sip:juliet@example.com>;tag=j89d
|  To: <sip:romeo@example.com>;tag=ffd2
|  Call-ID: 5BCF940D-793D-43F8-8972-218F7F4EAA8C
|  Event: presence
|  Max-Forwards: 70
|  CSeq: 42 SUBSCRIBE
|  Contact: <sip:juliet@example.com>;gr=yn0cl4bnw0yr3vym
|  Accept: application/pidf+xml
|  Expires: 0
|  Content-Length: 0
```

Upon receiving the SIP 200 OK acknowledging the cancellation, the XMPP-to-SIP gateway SHOULD send a <presence/> stanza of type "unsubscribed" addressed to the XMPP user:

Example 9: XMPP User Receives Unsubscribed Notification (F21)

```
|  <presence from='romeo@example.net'
|            to='juliet@example.com'
|            type='unsubscribed'/>
```

In accordance with Section 4.4.1 of [RFC6665], the XMPP-to-SIP gateway is then responsible for sending a NOTIFY message with a Subscription-State header of "terminated" in order to formally end the XMPP user's outbound presence authorization and the associated SIP dialog.

Example 10: XMPP-to-SIP Gateway Sends Presence Notification to Terminate Authorization (F25)

```
|  NOTIFY sip:juliet@example.com SIP/2.0
|  Via: SIP/2.0/TCP simple.example.net;branch=z9hG4bKna998sk
|  From: <sip:juliet@example.com>;tag=j89d
|  To: <sip:romeo@example.net>;tag=ffd2
|  Call-ID: 5BCF940D-793D-43F8-8972-218F7F4EAA8C
|  Event: presence
|  Subscription-State: terminated
|  Max-Forwards: 70
|  CSeq: 43 NOTIFY
|  Content-Length: 0
```

Note: When the XMPP user cancels her outbound presence authorization to the SIP user, any inbound authorization that she might have approved (thus enabling the SIP user to see her presence) remains unchanged.

5.3. SIP to XMPP

5.3.1. Requesting a Presence Authorization

The following diagram illustrates the protocol flow for establishing an authorization for a SIP user to receive presence notifications from an XMPP contact, as further explained in the text and examples after the diagram.

```
   SIP         SIP         XMPP         XMPP
   UA         Proxy       Server        Client
    |       + S2X GW         |           |
    |           |            |           |
    | (F26) SIP |            |           |
    | SUBSCRIBE |            |           |
    |**********>|            |           |
    | (F27) SIP |            |           |
    | 200 OK    |            |           |
    |<**********|            |           |
    |           | (F28) XMPP |           |
    |           | subscribe  |           |
    |           |...........>|           |
    |           |            | (F29) XMPP|
    |           |            | subscribe |
    |           |            |..........>|
    |           |            | (F30) XMPP|
    |           |            | subscribed|
    |           |            |<..........|
    |           | (F31) XMPP |           |
    |           | subscribed |           |
    |           |<...........|           |
    | (F32) SIP |            |           |
    | NOTIFY    |            |           |
    | (active)  |            |           |
    |<**********|            |           |
    | (F33) SIP |            |           |
    | 200 OK    |            |           |
    |**********>|            |           |
    |           |            |           |
```

A SIP User Agent initiates a presence authorization to an XMPP contact’s presence information by sending a SIP SUBSCRIBE request to the contact. The following is an example of such a request:

Example 11: SIP User Subscribes to XMPP Contact (F26)

```
|  SUBSCRIBE sip:juliet@example.com SIP/2.0
|  Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
|  From: <sip:romeo@example.net>;tag=xfg9
|  To: <sip:juliet@example.net>
|  Call-ID: AA5A8BE5-CBB7-42B9-8181-6230012B1E11
|  Event: presence
|  Max-Forwards: 70
|  CSeq: 1 SUBSCRIBE
|  Contact: <sip:romeo@example.net>;gr=dr4hcr0st3lup4c
|  Accept: application/pidf+xml
|  Content-Length: 0
```

Notice that the Expires header was not included in the SUBSCRIBE request; this means that the default value of 3600 (i.e., 3600 seconds = 1 hour) applies.

Upon receiving the SUBSCRIBE, the SIP proxy needs to determine the identity of the domain portion of the Request-URI, which it does by following the procedures explained in Section 5 of [RFC7247]. If the domain is an XMPP domain, the SIP proxy will hand off the SUBSCRIBE to an associated SIP-to-XMPP gateway or connection manager that natively communicates with XMPP servers.

The SIP-to-XMPP gateway is then responsible for translating the SUBSCRIBE into an XMPP authorization request addressed from the SIP user to the XMPP contact:

Example 12: XMPP Transformation of SIP SUBSCRIBE (F28)

```
|  <presence from=’romeo@example.net’
|            to=’juliet@example.com’
|            type=’subscribe’/>
```

In accordance with [RFC6121], the XMPP user’s server delivers the presence authorization request to the XMPP user (or, if an authorization already exists in the XMPP user’s roster, the XMPP server SHOULD auto-reply with a <presence/> stanza of type ’subscribed’).

The "happy path" is for the XMPP user to approve the presence authorization request by generating a <presence/> stanza of type "subscribed" (F30). The XMPP server then stamps that presence stanza with the 'from' address of the XMPP contact and sends it to the SIP user (F31). Upon receiving the stanza, the SIP-to-XMPP gateway generates an empty SIP NOTIFY message with a Subscription-State header [RFC6665] of "active", which serves to inform the SIP user that the presence authorization request has been approved (F32).

Example 13: XMPP User Approves Presence Authorization Request (F31)

```
|  <presence from='juliet@example.com'
|            to='romeo@example.net'
|            type='subscribed'/>
```

Example 14: Presence Authorization Request Approved (F32)

```
|  NOTIFY sip:romeo@example.net SIP/2.0
|  Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
|  From: <sip:romeo@example.net>;tag=xfg9
|  To: <sip:juliet@example.com>;tag=ur93
|  Call-ID: AA5A8BE5-CBB7-42B9-8181-6230012B1E11
|  Event: presence
|  Subscription-State: active
|  Max-Forwards: 70
|  CSeq: 2 NOTIFY
|  Content-Length: 0
```

As an alternative to the "happy path", the XMPP user could decline the presence authorization request by generating a <presence/> stanza of type "unsubscribed". The XMPP server would stamp that presence stanza with the 'from' address of the XMPP contact and would send it to the SIP user. The SIP-to-XMPP gateway then transforms that stanza into an empty SIP NOTIFY with a Subscription-State header [RFC6665] of "terminated" and a reason of "rejected":

Example 15: XMPP User Rejects Presence Authorization Request

```
|  <presence from='juliet@example.com'
|            to='romeo@example.net'
|            type='unsubscribed'/>
```

Example 16: Presence Authorization Request Rejected

```
|  NOTIFY sip:romeo@example.net SIP/2.0
|  Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
|  From: <sip:romeo@example.net>;tag=xfg9
|  To: <sip:juliet@example.com>;tag=ur93
|  Call-ID: AA5A8BE5-CBB7-42B9-8181-6230012B1E11
|  Event: presence
|  Subscription-State: terminated;reason=rejected
|  Max-Forwards: 70
|  CSeq: 2 NOTIFY
|  Content-Length: 0
```

5.3.2. Refreshing a Notification Dialog

For as long as a SIP user is online and wishes to maintain a notification session (i.e., receive presence notifications from the XMPP contact), the user’s SIP User Agent is responsible for periodically refreshing the notification dialog by sending an updated SUBSCRIBE request with an appropriate value for the Expires header. In response, the presence-aware SIP-to-XMPP gateway sends a SIP NOTIFY message to the SIP User Agent (per [RFC6665]); if the SIP-to-XMPP gateway has meaningful information about the availability state of the XMPP user (e.g., obtained from the core presence session in the XMPP server or learned by sending a presence probe as described under Section 7), then the NOTIFY communicates that information (e.g., by including a PIDF body [RFC3863] with the relevant data), whereas if the SIP-to-XMPP gateway does not have meaningful information about the availability state of the XMPP user, then the NOTIFY MUST be empty as allowed by [RFC6665].

5.3.3. Cancelling a Presence Authorization

SIP does not directly have a construct for cancelling an outbound presence authorization. Instead, the SIP user would terminate his outbound notification dialog by sending a SUBSCRIBE message whose Expires header is set to a value of zero ("0") and then never renew it:

Example 17: SIP User Terminates Notification Dialog

```
|  SUBSCRIBE sip:juliet@example.com SIP/2.0
|  Via: SIP/2.0/TCP simple.example.net;branch=z9hG4bKna998sk
|  From: <sip:romeo@example.net>;tag=xfg9
|  To: <sip:juliet@example.com>;tag=ur93
|  Call-ID: AA5A8BE5-CBB7-42B9-8181-6230012B1E11
|  Event: presence
|  Max-Forwards: 70
|  CSeq: 66 SUBSCRIBE
|  Contact: <sip:romeo@example.net>;gr=dr4hcr0st3lup4c
|  Expires: 0
|  Content-Length: 0
```

A presence-aware SIP-to-XMPP gateway is then responsible for:

1. Sending a SIP NOTIFY request to the SIP User Agent containing a PIDF document specifying that the XMPP contact now has a basic status of "closed", including a Subscription-State header [RFC6665] of "terminated" with a reason of "timeout".

2. Sending an XMPP <presence/> stanza of type "unavailable" to the XMPP contact.

Note: When the SIP user cancels his outbound presence authorization to the XMPP user, any inbound authorization that he might have approved (enabling the XMPP user to see his presence) remains unchanged.

6. Notifications of Presence Information

6.1. Overview

Both XMPP and presence-aware SIP systems enable entities (often, but not necessarily, human users) to send presence notifications to other entities. At its most basic, the term "presence" refers to information about an entity’s "on/off" availability for communication on a network. Often, this basic concept is supplemented by information that further specifies the entity’s context or status while available for communication; these availability states commonly include "away" and "do not disturb". Some systems and protocols extend the concepts of presence and availability even further and refer to any relatively ephemeral information about an entity as a kind of presence; categories of such "extended presence" include geographical location (e.g., GPS coordinates), user mood (e.g., grumpy), user activity (e.g., walking), and ambient environment (e.g., noisy). This document focuses on the "least common denominator" of network availability only. Future documents might address broader notions of presence, including availability states and extended presence or so-called "rich presence" as defined in specifications such as [RFC4480], [XEP-0107], and [XEP-0108].

The XMPP instant messaging and presence specification [RFC6121] defines how XMPP <presence/> stanzas can indicate availability (via the absence of a ’type’ attribute) or lack of availability (via a ’type’ attribute with a value of "unavailable"). SIP presence using a SIP event package for presence is specified in [RFC3856].

As described in [RFC6121], XMPP presence information about an entity is communicated by means of an XML <presence/> stanza sent over an XML stream. This document assumes that such a <presence/> stanza is sent from an XMPP client to an XMPP server over an XML stream negotiated between the client and the server, and that the client is controlled by a human user. In general, XMPP presence is sent by the user’s client to the user’s server and then broadcast to all entities who are subscribed to the user’s presence information.

As described in [RFC3856], presence information about an entity is communicated by means of a SIP NOTIFY event sent from a SIP User Agent to an intended recipient who is most generally referenced by a Presence URI of the form <pres:user@domain> but who might be referenced by a SIP or SIPS URI of the form <sip:user@domain> or <sips:user@domain>.

6.2. XMPP to SIP

When Juliet interacts with her XMPP client to modify her presence information (or when her client automatically updates her presence information, e.g., via an "auto-away" feature), her client generates an XMPP <presence/> stanza. The syntax of the <presence/> stanza, including required and optional elements and attributes, is defined in [RFC6121]. The following is an example of such a stanza:

Example 18: XMPP User Sends Presence Notification

```
|  <presence from='juliet@example.com/yn0cl4bnw0yr3vym'/>
```

Upon receiving such a stanza, the XMPP server to which Juliet has connected broadcasts it to all subscribers who are authorized to receive presence notifications from Juliet and who have indicated a current interest in receiving notifications (this is similar to the SIP NOTIFY method). For each subscriber, broadcasting the presence notification involves adding the 'to' address of the subscriber and then either delivering the notification to a local recipient (if the hostname in the subscriber's address matches one of the hostnames serviced by the XMPP server) or attempting to route it to the foreign domain that services the hostname in the subscriber's address. If the notification is bound for an address at a foreign domain, the XMPP server needs to determine the identity of the domainpart in the 'to' address, which it does by following the procedures discussed in [RFC7247]. If the domain is a SIP domain, the XMPP server will hand off the <presence/> stanza to an associated XMPP-to-SIP gateway or connection manager that natively communicates with presence-aware SIP proxy.

The XMPP-to-SIP gateway is then responsible for translating the XMPP <presence/> stanza into a SIP NOTIFY request (including the PIDF document) from the XMPP user to the SIP contact.

Example 19: SIP Transformation of XMPP Presence Notification

```
|  NOTIFY sip:juliet@example.com SIP/2.0
|  Via: SIP/2.0/TCP x2s.example.com;branch=z9hG4bKna998sk
|  From: <sip:juliet@example.com>;tag=gh19
|  To: <sip:romeo@example.net>
|  Contact: <sip:juliet@example.com>;gr=yn0cl4bnw0yr3vym
|  Call-ID: 2B44E147-3B53-45E4-9D48-C051F3216D14
|  Event: presence
|  Subscription-State: active;expires=599
|  Max-Forwards: 70
|  CSeq: 2 NOTIFY
|  Content-Type: application/pidf+xml
|  Content-Length: 192
|
|  <?xml version='1.0' encoding='UTF-8'?>
|  <presence xmlns='urn:ietf:params:xml:ns:pidf'
|            entity='pres:juliet@example.com'>
|    <tuple id='ID-yn0cl4bnw0yr3vym'>
|      <status>
|        <basic>open</basic>
|        <show xmlns='jabber:client'>away</show>
|      </status>
|    </tuple>
|  </presence>
```

The mapping of XMPP syntax elements to SIP syntax elements MUST be as shown in the following table. (Mappings for elements not mentioned are undefined and therefore are a matter of implementation.)

XMPP Element or Attribute	SIP Header or PIDF Data
<presence/> stanza	"Event: presence" (1)
XMPP resource identifier	tuple ’id’ attribute (2)
from	From
id	no mapping (3)
to	To
type	basic status (4) (5)
xml:lang	Content-Language
<priority/>	priority for tuple (6)
<show/>	no mapping (7)
<status/>	<note/>

Table 1: Presence Syntax Mapping from XMPP to SIP

Note the following regarding these mappings:

1. Only an XMPP <presence/> stanza that lacks a ’type’ attribute or whose ’type’ attribute has a value of "unavailable" is mapped by an XMPP-to-SIP gateway to a SIP NOTIFY request, because those are the only <presence/> stanzas that represent notifications.

2. The PIDF schema defines the tuple ’id’ attribute as having a datatype of "xs:ID"; because this datatype is more restrictive than the "xs:string" datatype for XMPP resourceparts (in particular, a number is not allowed as the first character of an ID), it is RECOMMENDED to prepend the resourcepart with "ID-" or some other alphabetic string when mapping from XMPP to SIP.

3. In practice, XMPP <presence/> stanzas often do not include the ’id’ attribute.

4. Because the lack of a ’type’ attribute indicates that an XMPP entity is available for communication, the XMPP-to-SIP gateway MUST map that information to a PIDF basic status of "open". Because a ’type’ attribute with a value of "unavailable" indicates that an XMPP entity is not available for communication, the XMPP-to-SIP gateway MUST map that information to a PIDF <basic/> status of "closed".

5. When the XMPP-to-SIP gateway receives an XMPP presence of type "unavailable" from the XMPP contact, it sends a SIP NOTIFY request from the XMPP contact to the SIP User Agent containing a PIDF document specifying that the XMPP contact now has a basic status of "closed".

6. The value of the XMPP <priority/> element is an integer between -128 and +127, whereas the value of the PIDF <contact/> element’s ’priority’ attribute is a decimal number from zero to one inclusive, with a maximum of three decimal places. If the value of the XMPP <priority/> element is negative, an XMPP-to-SIP gateway MUST NOT map the value. If an XMPP-to-SIP gateway maps positive values, it SHOULD treat XMPP priority 0 as PIDF priority 0 and XMPP priority 127 as PIDF priority 1, mapping intermediate values appropriately so that they are unique (e.g., XMPP priority 1 to PIDF priority 0.007, XMPP priority 2 to PIDF priority 0.015, and so on up through mapping XMPP priority 126 to PIDF priority 0.992; note that this is an example only and that the exact mapping is up to the implementation).

7. Some implementations support custom extensions to encapsulate detailed information about availability; however, there is no need to standardize a PIDF extension for this purpose, because PIDF is already extensible, and thus the XMPP <show/> element (qualified by the ’jabber:client’ namespace) can be included directly in the PIDF XML. The examples in this document illustrate this usage, which is RECOMMENDED. The most useful values are likely "away" and "dnd" (both defined in [RFC6121]), although note that in XMPP a value of "dnd" (short for "do not disturb") merely means "busy" and does not imply that a server or client ought to block incoming traffic while the user is in that state. Naturally, an XMPP-to-SIP gateway can choose to translate a custom extension into an established value of the XMPP <show/> element (as defined in [RFC6121]) or translate a <show/> element into a custom extension that the XMPP-to-SIP gateway knows is supported by the SIP User Agent of the intended recipient. Unfortunately, this behavior does not guarantee that information will not be lost; to help prevent information loss, an XMPP-to-SIP gateway ought to include both the <show/> element and the custom extension if it cannot suitably translate the custom value

into a <show/> value. However, there is no guarantee that the SIP receiver will render a standard XMPP <show/> value or custom extension.

In XMPP, a user can connect with multiple clients at the same time [RFC6120]; for presence notification purposes [RFC6121], each client is associated with a distinct resourcepart [RFC7622] and a contact’s SIP User Agent will receive a separate presence notification from each of the XMPP user’s clients. Although the interpretation of multiple presence notifications from a single user is a matter of implementation by the contact’s SIP User Agent, typically the SIP User Agent will show the "most available" status for the contact (e.g., if the user is online with three devices, one of which is "away", one of which is in "do not disturb" mode, and one of which is "available" with no qualifications, then the status shown will simply be "available"). In SIP, it is reasonable for a SIP User Agent to model multiple presence notifications from an XMPP user in the same way that it would handle multiple tuples from a SIP user.

6.3. SIP to XMPP

When Romeo changes his presence, his SIP User Agent generates a SIP NOTIFY request for any contacts that have presence authorizations and notification sessions. The syntax of the NOTIFY request is defined in [RFC3856]. The following is an example of such a request:

Example 20: SIP User Sends Presence Notification

```
|  NOTIFY sip:romeo@example.net SIP/2.0
|  Via: SIP/2.0/TCP simple.example.net;branch=z9hG4bKna998sk
|  From: <sip:romeo@example.net>;tag=yt66
|  To: <sip:juliet@example.com>;tag=bi54
|  Contact: <sip:romeo@example.net>;gr=dr4hcr0st3lup4c
|  Call-ID: C33C6C9D-0F4A-42F9-B95C-7CE86B526B5B
|  Event: presence
|  Subscription-State: active;expires=499
|  Max-Forwards: 70
|  CSeq: 8 NOTIFY
|  Content-Type: application/pidf+xml
|  Content-Length: 193
|
|  <?xml version='1.0' encoding='UTF-8'?>
|  <presence xmlns='urn:ietf:params:xml:ns:pidf'
|            entity='pres:romeo@example.net'>
|    <tuple id='ID-dr4hcr0st3lup4c'>
|      <status>
|        <basic>closed</basic>
|      </status>
|    </tuple>
|  </presence>
```

Upon receiving the NOTIFY, the SIP proxy needs to determine the identity of the domain portion of the Request-URI, which it does by following the procedures discussed in [RFC7247]. If the domain is an XMPP domain, the SIP proxy will hand off the NOTIFY to an associated SIP-to-XMPP gateway or connection manager that natively communicates with XMPP servers.

The SIP-to-XMPP gateway is then responsible for translating the NOTIFY into an XMPP <presence/> stanza addressed from the SIP user to the XMPP contact:

Example 21: XMPP Transformation of SIP Presence Notification

```
|  <presence from='romeo@example.net'
|            to='juliet@example.com/yn0cl4bnw0yr3vym'
|            type='unavailable'/>
```

The mapping of SIP syntax elements to XMPP syntax elements MUST be as shown in the following table. (Mappings for elements not mentioned are undefined and therefore are a matter of implementation.)

SIP Header or PIDF Data	XMPP Element or Attribute
basic status	type (1)
Content-Language	xml:lang
From	from
priority for tuple	<priority/> (2)
To	to
<note/>	<status/>
<show/>	<show/> (3)

Table 2: Presence Syntax Mapping from SIP to XMPP

Note the following regarding these mappings:

1. A PIDF basic status of "open" MUST be mapped to a <presence/> stanza with no ’type’ attribute, and a PIDF basic status of "closed" MUST be mapped to a <presence/> stanza whose ’type’ attribute has a value of "unavailable".

2. See the notes following Table 1 of this document regarding mapping of presence priority.

3. If a SIP implementation supports the XMPP <show/> element (qualified by the ’jabber:client’ namespace) as a PIDF extension for availability status as described in the notes following Table 1 of this document, the SIP-to-XMPP gateway is responsible for including that element in the XMPP presence notification.

7. Polling for Presence Information

Both SIP and XMPP provide methods for explicitly requesting one-time information about the current presence status of another entity. These are "polling" methods as opposed to the publish-subscribe methods described in the rest of this document.

7.1. XMPP to SIP

In XMPP, an explicit request for information about current presence status is completed by sending a <presence/> stanza of type "probe":

Example 22: XMPP Server Sends Presence Probe on Behalf of XMPP User

```
|  <presence from='juliet@example.com/chamber'
|            to='romeo@example.net'
|            type='probe'/>
```

Note: As described in [RFC6121], presence probes are used by XMPP servers to request presence on behalf of XMPP users; XMPP clients are discouraged from sending presence probes, because retrieving presence is a service that XMPP servers provide automatically.

A SIP-to-XMPP gateway would transform the presence probe into its SIP equivalent, which is a SUBSCRIBE request with an Expires header value of zero ("0") in a new dialog:

Example 23: SIP Transformation of XMPP Presence Probe

```
|  SUBSCRIBE sip:romeo@example.net SIP/2.0
|  Via: SIP/2.0/TCP x2s.example.com;branch=z9hG4bKna998sk
|  From: <sip:juliet@example.com>;tag=j89d
|  Call-ID: 2398B737-566F-4CBB-A21A-1F8EEF7AF423
|  Event: presence
|  Max-Forwards: 70
|  CSeq: 1 SUBSCRIBE
|  Contact: <sip:juliet@example.com>;gr=yn0cl4bnw0yr3vym
|  Accept: application/pidf+xml
|  Expires: 0
|  Content-Length: 0
```

As described in [RFC3856], this causes a NOTIFY to be sent to the subscriber, just as a presence probe does (the transformation rules for presence notifications have been previously described in Section 6.2 of this document).

7.2. SIP to XMPP

In SIP, an explicit request for information about current presence status is effectively completed by sending a SUBSCRIBE with an Expires header value of zero ("0"):

Example 24: SIP User Sends Presence Request

```
|  SUBSCRIBE sip:juliet@example.com SIP/2.0
|  Via: SIP/2.0/TCP simple.example.net;branch=z9hG4bKna998sk
|  From: <sip:romeo@example.net>;tag=yt66
|  Call-ID: 717B1B84-F080-4F12-9F44-0EC1ADE767B9
|  Event: presence
|  Max-Forwards: 70
|  CSeq: 1 SUBSCRIBE
|  Contact: <sip:romeo@example.net>;gr=dr4hcr0st3lup4c
|  Expires: 0
|  Content-Length: 0
```

A presence-aware SIP-to-XMPP gateway translates such a SIP request into a <presence/> stanza of type "probe" if it does not already have presence information about the contact:

Example 25: XMPP Transformation of SIP Presence Request

```
|  <presence from='romeo@example.net'
|            to='juliet@example.com'
|            type='probe'/>
```

8. Privacy and Security Considerations

Detailed privacy and security considerations are given for presence protocols in [RFC2779], for SIP-based presence in [RFC3856] (see also [RFC3261]), and for XMPP-based presence in [RFC6121] (see also [RFC6120]).

8.1. Amplification Attacks

There exists the possibility of an amplification attack launched from the XMPP network against a SIP Presence Server, because each long-lived XMPP presence authorization would typically result in multiple notification dialog refreshes on the SIP side of an XMPP-to-SIP gateway. Therefore, access to an XMPP-to-SIP gateway SHOULD be restricted in various ways; for example:

- Only an XMPP service that carefully controls account provisioning and provides effective methods for the administrators to control the behavior of registered users ought to host an XMPP-to-SIP gateway (e.g., not a service that offers open account registration).

- An XMPP-to-SIP gateway ought to be associated with only a single domain or trust realm. For example, an XMPP-to-SIP gateway hosted at simple.example.com ought to allow only users within the example.com domain to access the XMPP-to-SIP gateway, not users within example.org, example.net, or any other domain (unless they are part of the same multi-tenanted environment as example.com). This helps to prevent the gateway equivalent of open relays that are shared across XMPP domains from different trust realms.

If a SIP Presence Server receives communications through an XMPP-to-SIP gateway from users who are not associated with a domain that is so related to the hostname of the XMPP-to-SIP gateway, it SHOULD (based on local service provisioning) refuse to service such users or refuse to receive traffic from the XMPP-to-SIP gateway. As a further check, whenever an XMPP-to-SIP gateway seeks to refresh an XMPP user’s long-lived authorization to a SIP user’s presence, it first sends an XMPP <presence/> stanza of type "probe" from the address of the XMPP-to-SIP gateway to the "bare Jabber Identifier (JID)" (user@domain.tld) of the XMPP user, to which the user’s XMPP server responds in accordance with [RFC6121]; this puts an equal burden on the XMPP server and the SIP proxy.

8.2. Presence Leaks

Presence notifications can contain sensitive information (e.g., about network availability). In addition, it is possible in both SIP and XMPP for an entity to send different presence notifications to different subscribers. Therefore, a gateway MUST NOT route or deliver a presence notification to any entity other than the intended recipient (as represented by the ’to’ address for XMPP and by the Request-URI for SIP), because it does not possess information about authorization to receive presence notifications for such entities -- that information resides at the user’s home service, not at the receiving gateway.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[RFC3856] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, DOI 10.17487/RFC3856, August 2004, <http://www.rfc-editor.org/info/rfc3856>.

[RFC3857] Rosenberg, J., "A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)", RFC 3857, DOI 10.17487/RFC3857, August 2004, <http://www.rfc-editor.org/info/rfc3857>.

[RFC3863] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, DOI 10.17487/RFC3863, August 2004, <http://www.rfc-editor.org/info/rfc3863>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <http://www.rfc-editor.org/info/rfc6120>.

[RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, DOI 10.17487/RFC6121, March 2011, <http://www.rfc-editor.org/info/rfc6121>.

[RFC6665] Roach, A.B., "SIP-Specific Event Notification", RFC 6665, DOI 10.17487/RFC6665, July 2012, <http://www.rfc-editor.org/info/rfc6665>.

[RFC7247] Saint-Andre, P., Houri, A., and J. Hildebrand, "Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Architecture, Addresses, and Error Handling", RFC 7247, DOI 10.17487/RFC7247, May 2014, <http://www.rfc-editor.org/info/rfc7247>.

[RFC7622] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 7622, DOI 10.17487/RFC7622, September 2015, <http://www.rfc-editor.org/info/rfc7622>.

9.2. Informative References

[RFC2778] Day, M., Rosenberg, J., and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, DOI 10.17487/RFC2778, February 2000, <http://www.rfc-editor.org/info/rfc2778>.

[RFC2779] Day, M., Aggarwal, S., Mohr, G., and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, DOI 10.17487/RFC2779, February 2000, <http://www.rfc-editor.org/info/rfc2779>.

[RFC3860] Peterson, J., "Common Profile for Instant Messaging (CPIM)", RFC 3860, DOI 10.17487/RFC3860, August 2004, <http://www.rfc-editor.org/info/rfc3860>.

[RFC4480] Schulzrinne, H., Gurbani, V., Kyzivat, P., and J. Rosenberg, "RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF)", RFC 4480, DOI 10.17487/RFC4480, July 2006, <http://www.rfc-editor.org/info/rfc4480>.

[RFC4825] Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", RFC 4825, DOI 10.17487/RFC4825, May 2007, <http://www.rfc-editor.org/info/rfc4825>.

[RFC7572] Saint-Andre, P., Houri, A., and J. Hildebrand, "Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Instant Messaging", RFC 7572, DOI 10.17487/RFC7572, June 2015, <http://www.rfc-editor.org/info/rfc7572>.

[RFC7573] Saint-Andre, P. and S. Loreto, "Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): One-to-One Text Chat Sessions", RFC 7573, DOI 10.17487/RFC7573, June 2015, <http://www.rfc-editor.org/info/rfc7573>.

[RFC7702] Saint-Andre, P., Ibarra, S., and S. Loreto, "Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Groupchat", RFC 7702, DOI 10.17487/RFC7702, December 2015, <http://www.rfc-editor.org/info/rfc7702>.

[XEP-0107] Saint-Andre, P. and R. Meijer, "User Mood", XSF XEP 0107, October 2008, <http://xmpp.org/extensions/xep-0107.html>.

[XEP-0108] Meijer, R. and P. Saint-Andre, "User Activity", XSF XEP 0108, October 2008, <http://xmpp.org/extensions/xep-0108.html>.

Appendix A. Changes from RFC 7248

RFC 7248 had already been published when the STOX working group discovered that a related document (since published as [RFC7702]) contained problems that also applied to RFC 7248. Specifically, the diagrams and protocol flows in RFC 7248 contained errors that reflected an incorrect architecture with gateways on both sides of the protocol exchange; in theory and in practice, presence traffic from an XMPP system would be translated by an XMPP-to-SIMPLE gateway on the XMPP side and received by a normal SIP/SIMPLE system directly (without a receiving gateway on the SIP/SIMPLE side), and traffic from a SIP system would be translated by a SIMPLE-to-XMPP gateway on the SIP side and received by a normal XMPP system (without a receiving gateway on the XMPP side).

Therefore, this document makes the following substantive changes from RFC 7248:

- Corrects the architectural assumptions, diagrams, and protocol flows to reflect a single-gateway model in each direction.

- Adjusts terminology to replace the term "SIP Server" with the term "SIP Proxy" or "SIP Presence Server" as appropriate, and to use the term "notification dialog" for a SIP subscription and the term "presence authorization" for an XMPP subscription instead of the generic term "subscription" in both contexts.

- Clarifies that SIP notification dialogs are used to handle presence authorizations in SIP (e.g., there is no dedicated way to signal outbound cancellation of an authorization as there is in XMPP).

- Clarifies the use of the ’presence.winfo’ event package, of the SIP Subscription-State headers (specifically with values of "pending", "active", "closed", or "terminated"), and of SIP NOTIFY messages with no body.

- Clarifies the durations of notification dialogs and presence authorizations, and how they are extended in SIP and handled in XMPP.

- Removes the mapping of the XMPP ’id’ attribute to the SIP "CSeq" header.

- Describes the handling of multiple connected resources in XMPP.

- Provides information about mitigations for leaks of presence information.

Acknowledgements

Thanks to the authors, contributors, and other individuals acknowledged in RFC 7248.

Thanks to Saul Ibarra Corretge and Markus Isomaki for their reviews during working group consideration.

Special thanks to Ben Campbell for identifying the underlying discrepancy that resulted in the need to obsolete RFC 7248.

Thanks also to Markus Isomaki and Yana Stamcheva as the working group chairs and Alissa Cooper as the sponsoring Area Director.

Author’s Address

Peter Saint-Andre
Filament

Email: peter@filament.com
URI: https://filament.com/