Network Working Group   P. Saint-Andre
Internet-Draft   &yet
Intended status: Standards Track   A. Houri
Expires: August 13, 2014   IBM
                                                         J. Hildebrand
                                                    Cisco Systems, Inc.
                                                      February 9, 2014

**Interworking between the Session Initiation Protocol (SIP) and the
Extensible Messaging and Presence Protocol (XMPP): Presence
draft-ietf-stox-presence-09**

Abstract

   This document defines a bi-directional protocol mapping for the
   exchange of presence information between the Session Initiation
   Protocol (SIP) and the Extensible Messaging and Presence Protocol
   (XMPP).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

   o 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 helping ensure interworking between these protocols
   is to map each protocol to the abstract semantics described in
   [RFC3860]; although that is the approach taken by both [RFC3922] and
   [I-D.ietf-simple-cpim-mapping], to the best of our knowledge 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 to XMPP and vice-versa), since that is how
   existing systems solve the interworking problem.

   The architectural assumptions underlying such direct mappings are
   provided in [I-D.ietf-stox-core], 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.

2. Intended Audience

   The documents in this series 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 [I-D.ietf-stox-core], and with the
   following presence-related specifications:

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

   o Presence Information Data Format (PIDF) [RFC3863]

   o Extensible Messaging and Presence Protocol: Instant Messaging and
      Presence [RFC6121]

   o SIP-Specific Event Notification [RFC6665]

3. Terminology

   A number of terms used here (user, contact, subscription,
   notification, etc.) are explained in [RFC3261], [RFC3856], [RFC6120],
   and [RFC6121]. This document uses some, but not all, of the terms
   defined in the Model for Presence and Instant Messaging [RFC2778].

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. Subscriptions to Presence Information

4.1. 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 subscriptions are specified in [RFC6121].
Presence subscriptions using a SIP event package for presence are
specified in [RFC3856].

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

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

o "from" (the user has a subscription from the contact)

o "to" (the user has a subscription to the contact's presence
   information)

o "both" (both user and contact are subscribed to each other's
   presence information)

As described in [RFC3856], SIP presence subscriptions are managed
through the use of SIP SUBSCRIBE events 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>.
In practice, 'pres' URIs are rarely used, which is the examples in
this document use 'sip' URIs.

The subscription models underlying XMPP and SIP differ mainly in the
fact that XMPP presence subscriptions are long-lived (indeed
permanent if not explicitly cancelled, so that a subscription need
never be refreshed during any given presence "session"), whereas SIP
presence subscriptions are short-lived (the default time-to-live of a
SIP presence subscription is 3600 seconds, as specified in
Section 6.4 of [RFC3856], so that a subscription needs to be
explicitly refreshed if it will have the appearance of being
permanent or even of lasting as for the duration of a presence
"session"). This disparity has implications for the handling of

subscription cancellations in either direction and, from the SIP
side, subscription refreshes.

4.2. XMPP to SIP

4.2.1. Establishing a Presence Subscription

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

```
   XMPP          XMPP       XMPP-to-SIP    SIP-to-XMPP      SIP          SIP
   User         Server       Gateway        Gateway       Server        User
    |             |             |             |             |             |
    | (F1) XMPP   |             |             |             |             |
    | subscribe   |             |             |             |             |
    |---------->|             |             |             |             |
    |             | (F2) XMPP   |             |             |             |
    |             | subscribe   |             |             |             |
    |             |----------->|             |             |             |
    |             |             | (F3) SIP    |             |             |
    |             |             | SUBSCRIBE   |             |             |
    |             |             |------------>|             |             |
    |             |             |             | (F4) SIP    |             |
    |             |             |             | SUBSCRIBE   |             |
    |             |             |             |----------->|             |
    |             |             |             |             | (F5) SIP    |
    |             |             |             |             | SUBSCRIBE   |
    |             |             |             |             |---------->|
    |             |             |             |             | (F6) SIP    |
    |             |             |             |             | 200 OK      |
    |             |             |             | (F7) SIP    |<----------|
    |             |             |             | 200 OK      | (F8) SIP    |
    |             |             |             |<----------| NOTIFY      |
    |             |             |             |             |<----------|
    |             |             |             | (F9) SIP    |             |
    |             |             |             | NOTIFY      |             |
    |             |             |             |<----------|             |
    |             |             |             | (F10) SIP   |             |
    |             |             |             | 200 OK      |             |
    |             |             | (F12) XMPP  |----------->|             |
    |             |             | subscribed  |             | (F11) SIP   |
    |             |             |<------------|             | 200 OK      |
    |             | (F13) XMPP  |             |             |---------->|
    |             | subscribed  | (F15) XMPP  |             |             |
    |             |<----------| presence    |             |             |
    | (F14) XMPP|             |<------------|             |             |
    | subscribed| (F16) XMPP  |             |             |             |
    |<----------| presence    |             |             |             |
    |             |<----------|             |             |             |
    | (F17) XMPP|             |             |             |             |
    | presence   |             |             |             |             |
    |<----------|             |             |             |             |
```
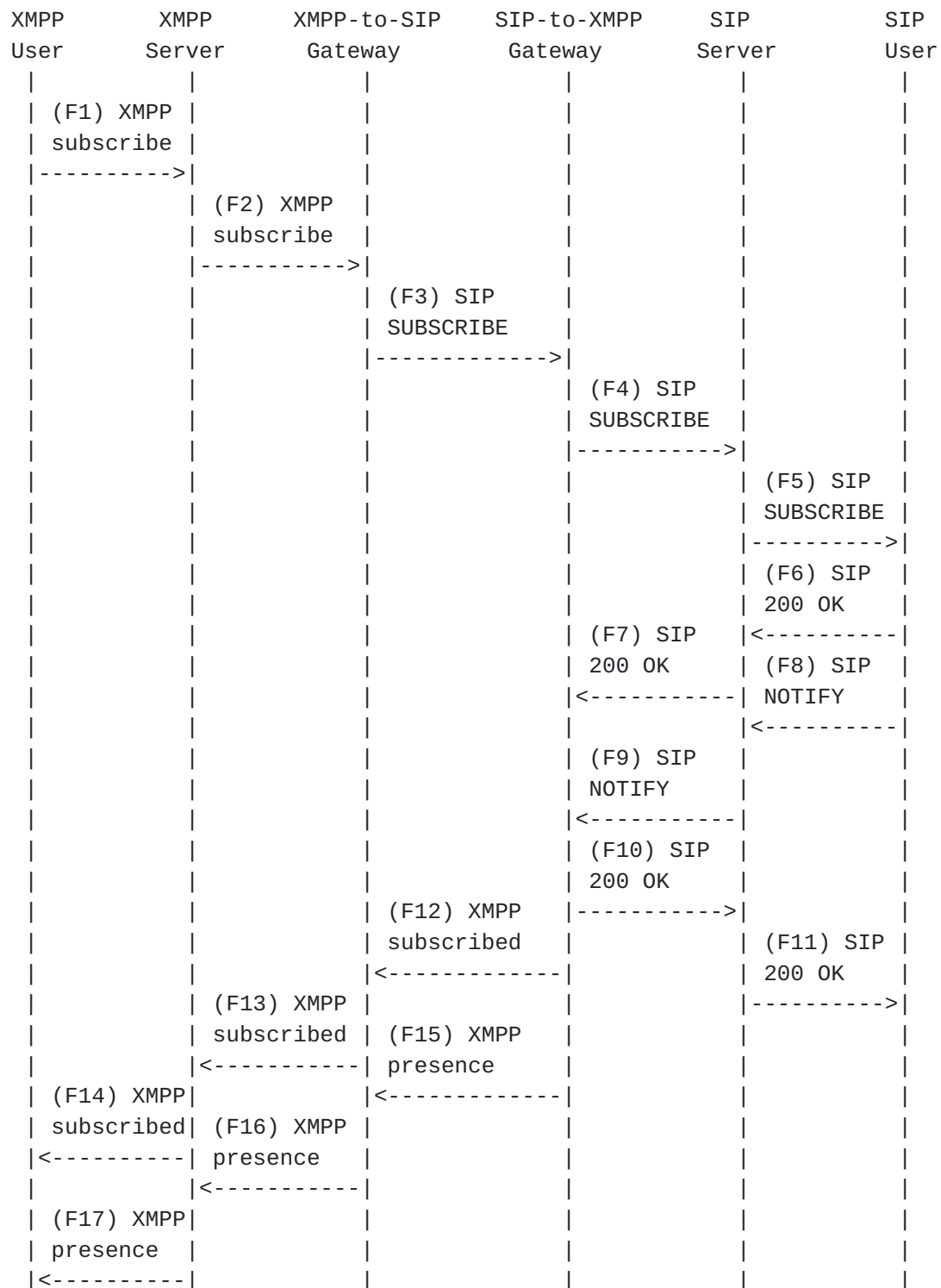
An XMPP user (e.g., juliet@example.com) initiates a subscription by
sending a subscription request to a contact (e.g.,
romeo@example.net), and the contact either accepts or declines the
request. If the contact accepts the request, the user will have a

subscription to the contact's presence information until (1) the user
unsubscribes or (2) the contact cancels the subscription.   The
subscription 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 [I-D.ietf-stox-core]. If the
domain is a SIP domain, the XMPP server will hand off the presence
stanza to an associated XMPP-SIP gateway or connection manager that
natively communicates with presence-aware SIP servers.

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

Example 2: SIP transformation of XMPP subscription request (F3)

```
|   SUBSCRIBE sip:romeo@example.net SIP/2.0
|   Via: SIP/2.0/TCP x2s.example.com;branch=z9hG4bKna998sk
|   From: <sip:juliet@example.com>;tag=ffd2
|   Call-ID: 5BCF940D-793D-43F8-8972-218F7F4EAA8C
|   Event: presence
|   Max-Forwards: 70
|   CSeq: 123 SUBSCRIBE
|   Contact: <sip:x2s.example.com;transport=tcp>
|   Accept: application/pidf+xml
|   Expires: 3600
|   Content-Length: 0
```

Once the SIP-XMPP gateway has passed the SIP SUBSCRIBE off to the SIP
server and the SIP server has delivered the SIP SUBSCRIBE to the SIP
user (F4 and F5, no examples shown), the SIP user then would send a
response indicating acceptance of the subscription request:

Example 3: SIP accepts subscription request (F6)

```
|  SIP/2.0 200 OK
|  Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
|  From: <sip:romeo@example.net>;tag=ffd2
|  To: <sip:juliet@example.com>;tag=j89d
|  Call-ID: 5BCF940D-793D-43F8-8972-218F7F4EAA8C
|  CSeq: 234 SUBSCRIBE
|  Contact: <sip:simple.example.net;transport=tcp>
|  Expires: 3600
|  Content-Length: 0
```

In accordance with [RFC6665], the XMPP-SIP gateway SHOULD consider
the subscription state to be "neutral" until it receives a NOTIFY
message. Therefore the SIP user or SIP-XMPP gateway at the SIP
user's domain SHOULD immediately send a NOTIFY message containing a
"Subscription-State" header whose value contains the string "active"
(see Section 5).

Example 4: SIP user sends presence notification (F7)

```
|  NOTIFY sip:192.0.2.1 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
|  Call-ID: 5BCF940D-793D-43F8-8972-218F7F4EAA8C
|  Event: presence
|  Subscription-State: active;expires=499
|  Max-Forwards: 70
|  CSeq: 8775 NOTIFY
|  Contact: <sip:simple.example.net;transport=tcp>
|  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-orchard'>
|      <status>
|        <basic>open</basic>
|        <show xmlns='jabber:client'>away</show>
|      </status>
|    </tuple>
|  </presence>
```

In response, the presence-aware SIP-XMPP gateway would send a 200 OK
to the SIP user (not shown here since it is not translated into an
XMPP stanza).

Upon receiving the first NOTIFY with a subscription state of active,
the XMPP-SIP gateway MUST generate 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 under Section 5, the gateway MUST also generate a
presence notification addressed to the XMPP user:

Example 6: XMPP user receives presence notification from SIP contact
(F17)

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

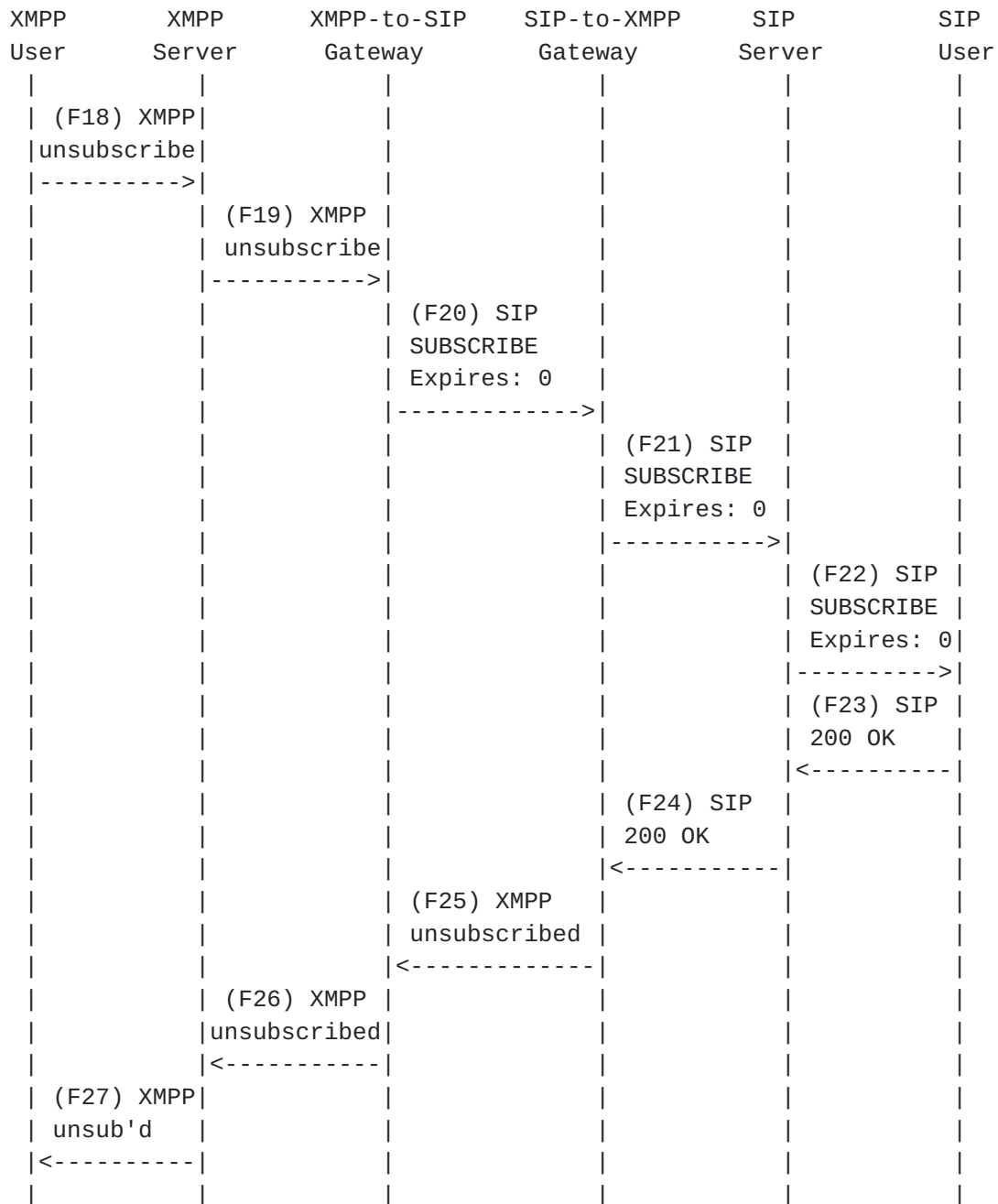## 4.2.2. Refreshing a Presence Subscription

It is the responsibility of the XMPP-SIP gateway to set the value of
the "Expires" header and to periodically renew the subscription on
the SIMPLE side of the gateway so that the subscription appears to be
permanent to the XMPP user. For example, the XMPP-SIP gateway SHOULD
send a new SUBSCRIBE request to the SIP user whenever the XMPP user
initiates a presence session with the XMPP server by sending initial
presence to its XMPP server. The XMPP-SIP gateway also SHOULD send a
new SUBSCRIBE request to the SIP user whenever the SIP presence
subscription 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 presence subscription are provided in
[RFC6665]. Those rules also apply to XMPP-SIP gateways.
Furthermore, an XMPP-SIP gateway MUST consider the XMPP subscription
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 maintain the long-lived XMPP presence
subscription. [RFC6665] explains more detailed considerations about
the handing of SIP responses in relation to subscription requests and
refreshes.

Finally, see the Security Considerations (Section 8) of this document
for important information and requirements regarding the security
implications of subscription refreshes.

4.2.3. Cancelling a Presence Subscription

   The following diagram illustrates the protocol flow for cancelling an
   XMPP user's presence subscription to a SIP user, as further explained
   in the text and examples after the diagram.

```
 XMPP          XMPP        XMPP-to-SIP     SIP-to-XMPP      SIP           SIP
 User         Server        Gateway         Gateway       Server        User
  |            |              |               |             |             |
  | (F18) XMPP|              |               |             |             |
  |unsubscribe|              |               |             |             |
  |---------->|              |               |             |             |
  |            | (F19) XMPP |               |             |             |
  |            | unsubscribe|               |             |             |
  |            |----------->|               |             |             |
  |            |              | (F20) SIP   |             |             |
  |            |              | SUBSCRIBE   |             |             |
  |            |              | Expires: 0  |             |             |
  |            |              |------------>|             |             |
  |            |              |               | (F21) SIP |             |
  |            |              |               | SUBSCRIBE |             |
  |            |              |               | Expires: 0 |            |
  |            |              |               |----------->|            |
  |            |              |               |             | (F22) SIP |
  |            |              |               |             | SUBSCRIBE |
  |            |              |               |             | Expires: 0|
  |            |              |               |             |---------->|
  |            |              |               |             | (F23) SIP |
  |            |              |               |             | 200 OK    |
  |            |              |               |             |<----------|
  |            |              |               | (F24) SIP |             |
  |            |              |               | 200 OK    |             |
  |            |              |               |<-----------|            |
  |            |              | (F25) XMPP  |             |             |
  |            |              | unsubscribed|             |             |
  |            |              |<------------|             |             |
  |            | (F26) XMPP |               |             |             |
  |            |unsubscribed|               |             |             |
  |            |<-----------|               |             |             |
  | (F27) XMPP|              |               |             |             |
  | unsub'd   |              |               |             |             |
  |<----------|              |               |             |             |
  |            |              |               |             |             |
```

   At any time after subscribing, the XMPP user can unsubscribe from the
   contact's presence. This is done by sending a presence stanza of
   type "unsubscribe":

   Example 7: XMPP user unsubscribes from SIP contact (F18)

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

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

   Example 8: SIP transformation of XMPP unsubscribe (F20)

   | 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
   | Call-ID: 9D9F00DF-FCA9-4E7E-B970-80B638D5218A
   | Event: presence
   | Max-Forwards: 70
   | CSeq: 789 SUBSCRIBE
   | Contact: <sip:x2s.example.com;transport=tcp>
   | Accept: application/pidf+xml
   | Expires: 0
   | Content-Length: 0

   Upon sending the transformed unsubscribe, the XMPP-SIP gateway SHOULD
   send a presence stanza of type "unsubscribed" addressed to the XMPP
   user:

   Example 9: XMPP user receives unsubscribed notification (F27)

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

4.3. SIP to XMPP

4.3.1. Establishing a Presence Subscription

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

```
   SIP         SIP       SIP-to-XMPP   XMPP-to-SIP      XMPP         XMPP
   User      Server        Gateway       Gateway       Server        User
    |           |             |             |             |            |
    | (F28) SIP |             |             |             |            |
    | SUBSCRIBE |             |             |             |            |
    |---------->|             |             |             |            |
    |           | (F29) SIP   |             |             |            |
    |           | SUBSCRIBE   |             |             |            |
    |           |----------->|              |             |            |
    |           |             | (F30) XMPP  |             |            |
    |           |             | subscribe   |             |            |
    |           |             |------------>|             |            |
    |           |             |             | (F31) XMPP  |            |
    |           |             |             | subscribe   |            |
    |           |             |             |----------->|             |
    |           |             |             |             | (F32) XMPP|
    |           |             |             |             | subscribe |
    |           |             |             |             |---------->|
    |           |             |             |             | (F33) XMPP|
    |           |             |             |             | subscribed|
    |           |             |             |             |<----------|
    |           |             |             | (F34) XMPP  |            |
    |           |             |             | subscribed  |            |
    |           |             |             |<-----------|             |
    |           |             | (F35) SIP   |             |            |
    |           |             | 200 OK      |             |            |
    |           |             |<------------|             |            |
    |           | (F36) SIP   |             |             |            |
    |           | 200 OK      |             |             |            |
    |           |<-----------|              |             |            |
    | (F37) SIP |             |             |             |            |
    | 200 OK    |             |             |             |            |
    |<----------|             |             |             |            |
    |           |             |             |             |            |
```
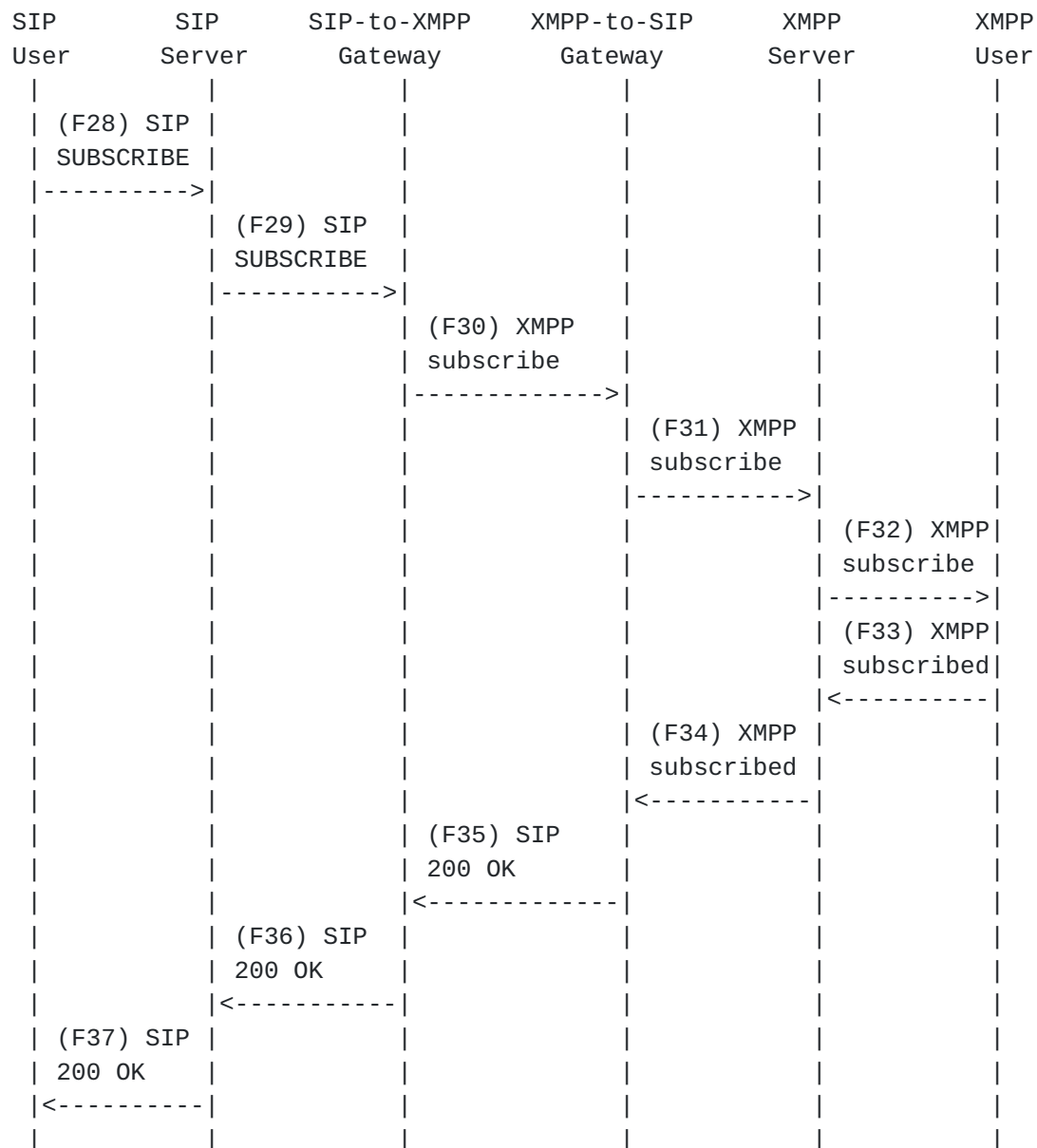
   A SIP user initiates a subscription to a contact's presence
   information by sending a SIP SUBSCRIBE request to the contact. The
   following is an example of such a request:

Example 10: SIP user subscribes to XMPP contact (F28)

```
|   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
|   Call-ID: AA5A8BE5-CBB7-42B9-8181-6230012B1E11
|   Event: presence
|   Max-Forwards: 70
|   CSeq: 263 SUBSCRIBE
|   Contact: <sip:simple.example.net;transport=tcp>
|   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 server needs to determine the identity of the domain portion of the Request-URI or To header, which it does by following the procedures explained in Section 5 of [I-D.ietf-stox-core]. If the domain is an XMPP domain, the SIP server will hand off the SUBSCRIBE to an associated SIP-XMPP gateway or connection manager that natively communicates with XMPP servers.

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

Example 11: XMPP transformation of SIP SUBSCRIBE (F30)

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

In accordance with [RFC6121], once it receives the stanza from the XMPP-SIP gateway, the XMPP user's server MUST deliver the presence subscription request to the XMPP user (or, if a subscription already exists in the XMPP user's roster, the XMPP server SHOULD auto-reply with a presence stanza of type 'subscribed').

If the XMPP user approves the subscription request, the XMPP server then MUST return a presence stanza of type "subscribed" addressed from the XMPP user to the SIP user. The XMPP-SIP gateway is responsible for translating the presence stanza of type "subscribed" into a SIP 200 OK response.

If the XMPP user declines the subscription request, the XMPP server then MUST return a presence stanza of type "unsubscribed" addressed

from the XMPP user to the SIP user and the XMPP-SIP gateway MUST
transform that stanza into an empty SIP NOTIFY message with a
Subscription-State of "terminated" and a reason of "rejected":

Example 12: Subscription request rejected

```
| NOTIFY sip:192.0.2.2 SIP/2.0
| Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
| From: <sip:juliet@example.com>;tag=ur93
| To: <sip:romeo@example.net>;tag=pq72
| Call-ID: AA5A8BE5-CBB7-42B9-8181-6230012B1E11
| Event: presence
| Subscription-State: terminated;reason=rejected
| Max-Forwards: 70
| CSeq: 232 NOTIFY
| Contact: <sip:x2s.example.com;transport=tcp>
| Content-Type: application/pidf+xml
| Content-Length: 0
```

4.3.2. Refreshing a Presence Subscription

For as long as a SIP user is online and interested in receiving
presence notifications from the XMPP contact, the user's SIP user
agent is responsible for periodically refreshing the subscription by
sending an updated SUBSCRIBE request with an appropriate value for
the Expires header. In response, the presence-aware SIP-XMPP gateway
MUST send a SIP NOTIFY to the user agent (per [RFC6665]); if the
gateway has meaningful information about the availability state of
the XMPP user (e.g., obtained from the core presence session in the
XMPP server) then the NOTIFY MUST communicate that information (e.g.,
by including a PIDF body [RFC3863] with the relevant data), whereas
if the gateway does not have meaningful information about the
availability state of the XMPP user then the NOTIFY MUST be empty as
allowed by [RFC6665].

Once the SIP user ends its presence session, it is the responsibility
of the presence-aware SIP-XMPP gateway to properly handle the
difference between short-lived SIP presence subscriptions and long-
lived XMPP presence subscriptions. The gateway has two options when
the SIP user's subscription expires:

o Cancel the subscription (i.e., treat it as temporary) and send an
   XMPP presence stanza of type "unsubscribe" to the XMPP contact;
   this honors the SIP semantic but will seem strange to the XMPP
   contact (since it will appear that the SIP user has cancelled a
   long-lived subscription).

   o Maintain the subscription (i.e., treat it as long-lived) and (1)
      send a SIP NOTIFY request to the SIP user containing a PIDF
      document specifying that the XMPP contact now has a basic status
      of "closed", including a Subscription-State of "terminated" with a
      reason of "timeout" and (2) send an XMPP presence stanza of type
      "unavailable" to the XMPP contact; this violates the letter of the
      SIP semantic but will seem more natural to the XMPP contact.

   Which of these options a presence-aware SIP-XMPP gateway chooses is
   up to the implementation.

   If the implementation chooses the first option, the protocol
   generated would be as follows:

   Example 13: XMPP handling of temporary subscription expiry

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

   If the implementation chooses the second option, the protocol
   generated would be as follows:

   Example 14: SIP handling of long-lived subscription expiry

   | NOTIFY sip:192.0.2.2 SIP/2.0
   | Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
   | From: <sip:juliet@example.com>;tag=ur93
   | To: <sip:romeo@example.net>;tag=pq72
   | Call-ID: 2B44E147-3B53-45E4-9D48-C051F3216D14
   | Event: presence
   | Subscription-State: terminated;reason=timeout
   | Max-Forwards: 70
   | CSeq: 232 NOTIFY
   | Contact: <sip:x2s.example.com;transport=tcp>
   | Content-Type: application/pidf+xml
   | Content-Length: 194
   |
   | <?xml version='1.0' encoding='UTF-8'?>
   | <presence xmlns='urn:ietf:params:xml:ns:pidf'
   | entity='pres:juliet@example.com'>
   | <tuple id='ID-balcony'>
   | <status>
   | <basic>closed</basic>
   | </status>
   | </tuple>
   | </presence>

Example 15: XMPP handling of long-lived subscription expiry

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

4.3.3. Cancelling a Presence Subscription

At any time, the SIP user can cancel the subscription by sending a
SUBSCRIBE message whose "Expires" header is set to a value of zero
("0"):

Example 16: SIP user cancels subscription

```
| 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: 8775 SUBSCRIBE
| Contact: <sip:simple.example.net;transport=tcp>
| Expires: 0
| Content-Length: 0
```

As above, upon receiving such a request, a presence-aware SIP-XMPP
gateway is responsible for doing one of the following:

o Cancel the subscription (i.e., treat it as temporary) and send an
   XMPP presence stanza of type "unsubscribe" to the XMPP contact.

o Maintain the subscription (i.e., treat it as long-lived) and (1)
   send a SIP NOTIFY request to the SIP user containing a PIDF
   document specifying that the XMPP contact now has a basic status
   of "closed", (2) send a SIP SUBSCRIBE request to the SIP user with
   an "Expires" header set to a value of "0" (zero) when it receives
   XMPP presence of type "unavailable" from the XMPP contact, and (3)
   send an XMPP presence stanza of type "unavailable" to the XMPP
   contact.

5. Notifications of Presence Information

5.1. 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). In this document, we focus on the "least common
denominator" of network availability only, although future documents
might address broader notions of presence, including availability
states and extended presence.

[RFC6121] defines how XMPP presence stanzas can indicate availability
(via 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. In this document we will assume 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 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 an
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>.

This document addresses basic presence or network availability only,
not the various extensions to SIP and XMPP for "rich presence", such
as [RFC4480], [XEP-0107], and [XEP-0108].

5.2. 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 17: XMPP user sends presence notification

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

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 (this is similar to the
SIP NOTIFY method). For each subscriber, broadcasting the presence
notification involves either delivering it 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. Thus
the XMPP server needs to determine the identity of the domainpart in
the 'to' address, which it does by following the procedures discussed
in [I-D.ietf-stox-core]. If the domain is a SIP domain, the XMPP
server will hand off the presence stanza to an associated XMPP-SIP
gateway or connection manager that natively communicates with
presence-aware SIP servers (F2, no example shown).

The XMPP-SIP gateway is then responsible for translating the XMPP
presence stanza into a SIP NOTIFY request and included PIDF document
from the XMPP user to the SIP user.

Example 18: SIP transformation of XMPP presence notification

```
| NOTIFY sip:192.0.2.2 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>;tag=yt66
| Contact: <sip:juliet@example.com>;gr=balcony
| Call-ID: 2B44E147-3B53-45E4-9D48-C051F3216D14
| Event: presence
| Subscription-State: active;expires=599
| Max-Forwards: 70
| CSeq: 157 NOTIFY
| Contact: <sip:x2s.example.com;transport=tcp>
| 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-balcony'>
|     <status>
|       <basic>open</basic>
|       <show xmlns='jabber:client'>away</show>
|     </status>
|   </tuple>
| </presence>
```

The mapping of XMPP syntax elements to SIP syntax elements SHOULD be
as shown in the following table. (Mappings for elements not
mentioned are undefined.)

Table 1: Presence syntax mapping from XMPP to SIP

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

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" SHOULD be
    mapped by an XMPP-SIP gateway to a SIP NOTIFY request, since
    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, often XMPP presence stanzas do not include the 'id'
    attribute.

4. Because the lack of a 'type' attribute indicates that an XMPP
    entity is available for communications, the gateway SHOULD 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 communications, the gateway SHOULD
   map that information to a PIDF <basic/> status of "closed".

5. When the XMPP-SIP gateway receives XMPP presence of type
   "unavailable" from the XMPP contact, it SHOULD (1) send a SIP
   NOTIFY request to the SIP user containing a PIDF document
   specifying that the XMPP contact now has a basic status of
   "closed" and (2) send a SIP SUBSCRIBE request to the SIP user
   with an "Expires" header set to a value of "0" (zero).

6. The value of the XMPP <priority/> element is an integer between
   -128 and +127, whereas the the value of the PIDF
   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 element is negative, an XMPP-SIP
   gateway MUST NOT map the value. If an XMPP-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
   this information; however, there is no need to standardize a PIDF
   extension for this purpose, since PIDF is already extensible and
   thus the <show/> element can be included directly, qualified by
   the 'jabber:client' namespace in the PIDF XML. The examples in
   this document illustrate this usage, which is RECOMMENDED. The
   most useful values are likely "away" and "dnd", although note
   that the latter value merely means "busy" and does not imply that
   a server or client ought to block incoming traffic while the user
   is in that state.

8. Some implementations support custom extensions to encapsulate
   detailed information about availability; however, there is no
   need to standardize a PIDF extension for this purpose, since PIDF
   is already extensible and thus the <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", although note that the latter value 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, a
   gateway can choose to translate a custom extension into an
   established value of the <show/> element [RFC6121], or translate
   a <show/> element into a custom extension that the gateway knows

      is supported by the user agent of the intended recipient.
      Unfortunately, this behavior does not guarantee that information
      will not be lost; to help prevent information loss, a gateway
      ought to include both the <show/> element and the custom
      extension if the gateway cannot suitably translate the custom
      value into a <show/> value.

5.3. SIP to XMPP

   When Romeo changes his presence, his SIP user agent generates a SIP
   NOTIFY request for any active subscriptions. The syntax of the
   NOTIFY request is defined in [RFC3856]. The following is an example
   of such a request:

   Example 19: SIP user sends presence notification

   | NOTIFY sip:192.0.2.1 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=orchard
   | Call-ID: C33C6C9D-0F4A-42F9-B95C-7CE86B526B5B
   | Event: presence
   | Subscription-State: active;expires=499
   | Max-Forwards: 70
   | CSeq: 8775 NOTIFY
   | Contact: <sip:simple.example.net;transport=tcp>
   | 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-orchard'>
   | <status>
   | <basic>closed</basic>
   | </status>
   | </tuple>
   | </presence>

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

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

Example 20: XMPP transformation of SIP presence notification

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

The mapping of SIP syntax elements to XMPP syntax elements SHOULD be
as shown in the following table. (Mappings for elements not
mentioned are undefined.)

Table 2: Presence syntax mapping from SIP to XMPP

SIP Header or PIDF Data	XMPP Element or Attribute
basic status	type (1)
Content-Language	xml:lang
CSeq	id (2)
From	from
priority for tuple	<priority/> (3)
To	to
<note/>	<status/>
<show/>	<show/> (4)

Note the following regarding these mappings:

1. A PIDF basic status of "open" SHOULD be mapped to no 'type'
    attribute, and a PIDF basic status of "closed" SHOULD be mapped
    to a 'type' attribute whose value is "unavailable".

2. This mapping is OPTIONAL.

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

   4. If a SIP implementation supports the <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-XMPP gateway is responsible for
       including that element in the XMPP presence notification.

6. Requests for Presence Information

   Both SIP and XMPP provide methods for requesting presence information
   about another entity.

6.1. XMPP to SIP

   In XMPP, a request for presence information is completed by sending a
   presence stanza of type "probe":

   Example 21: 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 since retrieving presence is
   a service that servers provide.

   An XMPP-SIP gateway would transform the presence probe into its SIP
   equivalent, which is a SUBSCRIBE request with an Expires header value
   of zero:

   Example 22: 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=ffd2
   | Call-ID: 5BCF940D-793D-43F8-8972-218F7F4EAA8C
   | Event: presence
   | Max-Forwards: 70
   | CSeq: 123 SUBSCRIBE
   | Contact: <sip:x2s.example.com;transport=tcp>
   | Accept: application/pidf+xml
   | Expires: 0
   | Content-Length: 0

   As described in [RFC3856], this cancels any subscription but 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 this document).

6.2. SIP to XMPP

In SIP, a request for presence information is effectively completed
by sending a SUBSCRIBE with an Expires header value of zero:

Example 23: 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: 8775 SUBSCRIBE
| Contact: <sip:simple.example.net;transport=tcp>
| Expires: 0
| Content-Length: 0
```

When honoring the long-lived semantics of an XMPP presence
subscription, a presence-aware SIP-XMPP gateway SHOULD translate such
a SIP request into a presence stanza of type 'probe' if it does not
already have presence information about the subscribee:

Example 24: XMPP transformation of SIP presence request

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

7. IANA Considerations

This document makes no requests of IANA.

8. Security Considerations

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

The mismatch between long-lived XMPP presence subscriptions and
short-lived SIP presence subscriptions introduces the possibility of
an amplification attack launched from the XMPP network against a SIP
presence server (since each long-lived XMPP presence subscription
would typically result in multiple subscription refresh requests on
the SIP side of a gateway). Therefore, access to an XMPP-SIP gateway

SHOULD be restricted in various ways; among other things, 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 such a gateway (e.g., not
a service that offers open account registration) and a gateway ought
to be associated only with a single domain or trust realm (e.g., a
gateway hosted at simple.example.com ought to allow only users within
the example.com domain to access the gateway, not users within
example.org, example.net, or any other domain). If a SIP presence
server receives communications through an XMPP-SIP gateway from users
who are not associated with a domain that is so related to the
hostname of the gateway, it SHOULD (based on local service
provisioning) refuse to service such users or refuse to receive
traffic from with the gateway. As a futher check, whenever an XMPP-
SIP gateway seeks to refresh an XMPP user's long-lived subscription
to a SIP user's presence, it MUST first send an XMPP
stanza of type "probe" from the address of the gateway to the "bare
JID" (user@domain.tld) of the XMPP user, to which the user's XMPP
server MUST respond in accordance with [RFC6121]; this puts an equal
burden on the XMPP server as on the SIP server.

9. References

9.1. Normative References

[I-D.ietf-stox-core]
           Saint-Andre, P., Houri, A., and J. Hildebrand,
           "Interworking between the Session Initiation Protocol
           (SIP) and the Extensible Messaging and Presence Protocol
           (XMPP): Core", draft-ietf-stox-core-09 (work in progress),
           December 2013.

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

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
           A., Peterson, J., Sparks, R., Handley, M., and E.
           Schooler, "SIP: Session Initiation Protocol", RFC 3261,
           June 2002.

[RFC3856] Rosenberg, J., "A Presence Event Package for the Session
           Initiation Protocol (SIP)", RFC 3856, August 2004.

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

   [RFC6120]   Saint-Andre, P., "Extensible Messaging and Presence
               Protocol (XMPP): Core", RFC 6120, March 2011.

   [RFC6121]   Saint-Andre, P., "Extensible Messaging and Presence
               Protocol (XMPP): Instant Messaging and Presence", RFC
               6121, March 2011.

   [RFC6665]   Roach, A., "SIP-Specific Event Notification", RFC 6665,
               July 2012.

9.2. Informative References

   [I-D.ietf-simple-cpim-mapping]
               Rosenberg, J. and B. Campbell, "CPIM Mapping of SIMPLE
               Presence and Instant Messaging", draft-ietf-simple-cpim-
               mapping-01 (work in progress), June 2002.

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

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

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

   [RFC3922]   Saint-Andre, P., "Mapping the Extensible Messaging and
               Presence Protocol (XMPP) to Common Presence and Instant
               Messaging (CPIM)", RFC 3922, October 2004.

   [RFC4480]   Schulzrinne, H., Gurbani, V., Kyzivat, P., and J.
               Rosenberg, "RPID: Rich Presence Extensions to the Presence
               Information Data Format (PIDF)", RFC 4480, July 2006.

   [XEP-0107]
               Saint-Andre, P. and R. Meijer, "User Mood", XSF XEP 0107,
               October 2008.

   [XEP-0108]
               Meijer, R. and P. Saint-Andre, "User Activity", XSF XEP
               0108, October 2008.

Appendix A. Acknowledgements

   The authors wish to thank the following individuals for their
   feedback: Chris Christou, Fabio Forno, Adrian Georgescu, Philipp
   Hancke, Saul Ibarra Corretge, Markus Isomaki, Olle Johansson, Paul

Authors' Addresses

      Peter Saint-Andre
      &yet

      Email: ietf@stpeter.im

      Avshalom Houri
      IBM
      Rorberg Building, Pekris 3
      Rehovot 76123
      Israel

      Email: avshalom@il.ibm.com

      Joe Hildebrand
      Cisco Systems, Inc.
      1899 Wynkoop Street, Suite 600
      Denver, CO 80202
      USA

      Email: jhildebr@cisco.com