Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2014

P. Saint-Andre
Cisco Systems, Inc.
S. Loreto
E. Gavita
N. Hossain
Ericsson

July 1, 2013

Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): One-to-One Text Chat

draft-ietf-stox-chat-00

Abstract

This document defines a bidirectional protocol mapping for the exchange of instant messages in the context of a one-to-one chat session between a user of the Session Initiation Protocol (SIP) and a user of the Extensible Messaging and Presence Protocol (XMPP). Specifically for SIP text chat, this document specifies a mapping to the Message Session Relay Protocol (MSRP).

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
3. XMPP to MSRP 4
4. MSRP to XMPP 7
5. Security Considerations 10
6. IANA Considerations 11
7. References 11
7.1. Normative References 11
7.2. Informative References 12
Appendix A. Acknowledgements 12
Authors' Addresses 12

1. Introduction

Both the Session Initiation Protocol [RFC3261] and the Extensible Messaging and Presence Protocol [RFC6120] can be used for the purpose of one-to-one text chat over the Internet. To ensure interworking between these technologies, it is important to define bidirectional protocol mappings.

The architectural assumptions underlying such protocol mappings are provided in [I-D.ietf-stox-core], including mapping of addresses and error conditions. This document specifies mappings for one-to-one text chat sessions (sometimes called "session-mode" messaging); in particular, this document specifies mappings between XMPP messages of type "chat" and the Message Session Relay Protocol [RFC4975]. Mappings for single instant messages and groupchat are provided in separate documents.

The approach taken here is to directly map syntax and semantics from one protocol to another. The mapping described herein depends on the protocols defined in the following specifications:

- o XMPP chat sessions using message stanzas of type "chat" are specified in [RFC6121].
- o SIP-based chat sessions using the SIP INVITE and SEND request types are specified in [RFC4975].

In SIMPLE, a chat session is formally negotiated just as any other session type is using SIP. By contrast, a one-to-one chat "session" in XMPP is an informal construct and is not formally negotiated: a user simply sends a message of type "chat" to a contact, the contact then replies to the message, and the sum total of such messages exchanged during a defined period of time is considered to be a chat session. To overcome the disparity between these approaches, a gateway that wishes to map between SIP and XMPP for one-to-one chat sessions needs to maintain some additional state, as described below.

The discussion venue for this document is the mailing list of the STOX WG; visit https://www.ietf.org/mailman/listinfo/stox for subscription information and discussion archives.

2. Terminology

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].

3. XMPP to MSRP

In XMPP, the "informal session" approach is to simply send someone a <message/> of type "chat" without starting any session negotiation ahead of time (as described in [RFC6121]). The XMPP "informal session" approach maps very well into a SIP MESSAGE request, as described in [I-D.ietf-stox-core]. However, the XMPP informal session approach can also be mapped to MSRP if the XMPP-to-SIP gateway maintains additional state.

The order of events is as follows.

```
XMPP User                       GW                       SIP User
    |                           |                           |
    |(F1) (XMPP) Chat message   |                           |
    |-------------------------->|                           |
    |                           |(F2) (SIP) INVITE          |
    |                           |-------------------------->|
    |                           |(F3) (SIP) 200 OK          |
    |                           |<--------------------------|
    |                           |(F4) (SIP) ACK             |
    |                           |-------------------------->|
    |                           |(F5) (MSRP) SEND           |
    |                           |-------------------------->|
    |                           |(F6) (MSRP) A reply        |
    |                           |<--------------------------|
    |(F7) (XMPP) A reply        |                           |
    |<--------------------------|                           |
    |                           |                           |
    .                           .                           .
    .                           .                           .
    .                           .                           .
    |                           |                           |
    |                           |(F8) (SIP) BYE             |
    |                           |<--------------------------|
    |                           |(F9) (SIP) 200 OK          |
    |                           |-------------------------->|
    |                           |                           |
```

First the XMPP user would generate an XMPP chat message.

Example: (F1) Juliet sends an XMPP message

```
<message from='juliet@example.com/balcony'
         to='romeo@example.net'
         type='chat'>
  <thread>711609sa</thread>
  <body>Art thou not Romeo, and a Montague?</body>
</message>
```

The local SIP-to-XMPP gateway at the SIMPLE server would then determine if Romeo supports MSRP. If so, the SIP-to-XMPP gateway would initiate an MSRP session with Romeo on Juliet's behalf.

Example: (F2) Gateway starts a formal session on behalf of Juliet

```
INVITE sip:romeo@example.net SIP/2.0
To: <sip:romeo@example.net>
From: <sip:juliet@example.com>
Contact: <sip:juliet@example.com>;gr=balcony
Subject: Open chat with Juliet?
Call-ID: 711609sa
Content-Type: application/sdp

c=IN IP4 x2s.example.com
m=message 7654 TCP/MSRP *
a=accept-types:text/plain
a=lang:en
a=lang:it
a=path:msrp://x2s.example.com:7654/jshA7weztas;tcp
```

Here we assume that Romeo accepts the MSRP session request.

Example: (F3) Romeo accepts the request

```
SIP/2.0 200 OK
To: <sip:juliet@example.com>;gr=balcony
From: <sip:romeo@example.net>
Contact: <sip:romeo@example.net>;gr=orchard
Call-ID: 711609sa
Content-Type: application/sdp

c=IN IP4 s2x.example.net
m=message 12763 TCP/MSRP *
a=accept-types:text/plain
a=lang:it
a=path:msrp://s2x.example.net:12763/kjhd37s2s20w2a;tcp
```

The XMPP-to-SIP gateway then acknowledges the session acceptance on

behalf of Romeo.

Example: (F4) Gateway sends ACK to Romeo’s UA

```
ACK sip:juliet@example.com SIP/2.0
To: <sip:romeo@example.net>;gr=orchard
From: <sip:juliet@example.com>
Contact: <sip:juliet@example.com>;gr=balcony
Call-ID: 711609sa
```

The XMPP-to-SIP gateway then transforms the original XMPP chat message into MSRP.

Example: (F5) Gateway transforms XMPP message to MSRP

```
MSRP a786hjs2 SEND
From-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
To-Path: msrp://s2x.example.net:12763/kjhd37s2s20w2a;tcp
Message-ID: 87652491
Byte-Range: 1-25/25
Content-Type: text/plain

Art thou not Romeo, and a Montague?
-------a786hjs2$
```

Romeo can then send a reply using his MSRP user agent.

Example: (F6) Romeo sends a reply

```
MSRP a786hjs2 SEND
To-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
From-Path: msrp://s2x.example.net:12763/kjhd37s2s20w2a;tcp
Message-ID: 87652491
Byte-Range: 1-25/25
Failure-Report: no
Content-Type: text/plain

Neither, fair saint, if either thee dislike.
-------a786hjs2$
```

The SIP-to-XMPP gateway would then transform that message into appropriate XMPP syntax for routing to the intended recipient.

Example: (F7) Gateway transforms MSRP message to XMPP

```
<message from='romeo@example.net/orchard'
         to='juliet@example.com/balcony'
         type='chat'>
  <thread>711609sa</thread>
  <body>Neither, fair saint, if either thee dislike.</body>
</message>
```

When the MSRP user wishes to end the chat session, the user's MSRP client sends a SIP BYE.

Example: (F8) Romeo terminates the chat session

```
BYE juliet@example.com sip: SIP/2.0
Max-Forwards: 70
From: <sip:romeo@example.net>;tag=087js
To: <sip:juliet@example.com>;tag=786
Call-ID: 711609sa
Cseq: 1 BYE
Content-Length: 0
```

The BYE is then acknowledged by the XMPP-to-SIP gateway.

Example: (F9) Gateway acknowledges termination

```
SIP/2.0 200 OK
From: <sip:juliet@example.com>;tag=786
To: <sip:romeo@example.net>;tag=087js
Call-ID: 711609sa
CSeq: 1 BYE
Content-Length: 0
```

4. MSRP to XMPP

When an MSRP client sends messages through a gateway to an XMPP client that does not support formal sessinos, the order of events is as follows.

```
   SIP User                     GW                      XMPP User
      |                         |                           |
      |(F1)(SIP) INVITE         |                           |
      |------------------------>|                           |
      |(F2)(SIP) 200 OK         |                           |
      |<------------------------|                           |
      |(F3)(SIP) ACK            |                           |
      |------------------------>|                           |
      |(F4)(MSRP) SEND          |                           |
      |------------------------>|                           |
      |                         |(F5)(XMPP) A chat message  |
      |                         |-------------------------->|
      |                         |(F6)(XMPP) A reply         |
      |                         |<--------------------------|
      |                         |                           |
      |(F7)(MSRP) SEND          |                           |
      |<------------------------|                           |
      |                         |                           |
      .                         .                           .
      .                         .                           .
      .                         .                           .
      |                         |                           |
      |(F8)(SIP) BYE            |                           |
      |------------------------>|                           |
      |(F9)(SIP) 200 OK         |                           |
      |<------------------------|                           |
      |                         |                           |
```

Example: (F1) SIP user starts the session

```
INVITE sip:juliet@example.com SIP/2.0
To: <sip:juliet@example.com>
From: <sip:romeo@example.net>
Contact: <sip:romeo@example.net>;gr=orchard
Subject: Open chat with Romeo?
Call-ID: 742507no
Content-Type: application/sdp

c=IN IP4 s2x.example.net
m=message 7313 TCP/MSRP *
a=accept-types:text/plain
a=lang:en
a=lang:it
a=path:msrp://s2x.example.net:7313/ansp71weztas;tcp
```

Example: (F2) Gateway accepts session on Juliet's behalf

```
SIP/2.0 200 OK
To: <sip:romeo@example.net>;gr=orchard
From: <sip:juliet@example.com>
Contact: <sip:juliet@example.com>;gr=balcony
Call-ID: 742507no
Content-Type: application/sdp

c=IN IP4 x2s.example.com
m=message 8763 TCP/MSRP *
a=accept-types:text/plain
a=lang:it
a=path:msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
```

Example: (F3) Romeo sends ACK

```
ACK sip:juliet@example.com SIP/2.0
To: <sip:juliet@example.com>;gr=balcony
From: <sip:romeo@example.net>
Contact: <sip:romeo@example.net>;gr=orchard
Call-ID: 742507no
```

Example: (F4) Romeo sends a message

```
MSRP ad49kswow SEND
To-Path: msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
From-Path: msrp://s2x.example.net:7313/ansp71weztas;tcp
Message-ID: 44921zaqwsx
Byte-Range: 1-32/32
Failure-Report: no
Content-Type: text/plain

I take thee at thy word ...
-------ad49kswow$
```

Example: (F5) Romeo sends a message (XMPP translation)

```
<message from='romeo@example.net'
         to='juliet@example.com'
         type='chat'>
  <thread>742507no</thread>
  <body>I take thee at thy word ...</body>
</message>
```

Example: (F6) Juliet sends a reply

```
<message from='juliet@example.com'
         to='romeo@example.net'
         type='chat'>
  <thread>711609sa</thread>
  <body>What man art thou ...?</body>
</message>
```

Example: (F8) Gateway transforms XMPP message to MSRP

```
MSRP a786hjs2 SEND
To-Path: msrp://s2x.example.net:7313/jshA7weztas;tcp
From-Path: msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
Message-ID: 87652491
Byte-Range: 1-25/25
Failure-Report: no
Content-Type: text/plain

What man art thou ...?
-------a786hjs2$
```

Example: (F9) Romeo terminates the session

```
BYE juliet@example.com sip: SIP/2.0
Max-Forwards: 70
To: <sip:juliet@example.com>;gr=balcony
From: <sip:romeo@example.net>
Contact: <sip:romeo@example.net>;gr=orchard
Call-ID: 742507no
Cseq: 1 BYE
Content-Length: 0
```

Example: (F10) Gateway acknowledges the termination of the session on behalf of XMPP user

```
SIP/2.0 200 OK
To: <sip:juliet@example.com>;gr=balcony
From: <sip:romeo@example.net>
Contact: <sip:romeo@example.net>;gr=orchard
Call-ID: 742507no
CSeq: 1 BYE
```

5. Security Considerations

Detailed security considerations for instant messaging protocols are given in [RFC2779], for SIP-based instant messaging in [RFC3428] (see

also [RFC3261]), and for XMPP-based instant messaging in [RFC6121] (see also [RFC6120]).

This document specifies methods for exchanging instant messages through a gateway that translates between SIP and XMPP. Such a gateway MUST be compliant with the minimum security requirements of the instant messaging protocols for which it translates (i.e., SIP and XMPP). The addition of gateways to the security model of instant messaging specified in [RFC2779] introduces some new risks. In particular, end-to-end security properties (especially confidentiality and integrity) between instant messaging user agents that interface through a SIMPLE-XMPP gateway can be provided only if common formats are supported. Specification of those common formats is out of scope for this document, although it is recommended to use [RFC3862] for instant messages.

6. IANA Considerations

This document requests no actions of IANA.

7. References

7.1. 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-00 (work in progress), July 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.

[RFC3862] Klyne, G. and D. Atkins, "Common Presence and Instant Messaging (CPIM): Message Format", RFC 3862, August 2004.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[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.

7.2. Informative References

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

[RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

Appendix A. Acknowledgements

Some text in this document was borrowed from [I-D.ietf-stox-core].

Thanks to Adrian Georgescu, Saul Ibarra, and Tory Patnoe for their feedback.

Authors' Addresses

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3282
Email: psaintan@cisco.com

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Salvatore.Loreto@ericsson.com

Eddy Gavita
Ericsson
Decarie Boulevard
Town of Mount Royal, Quebec
Canada

Email: eddy.gavita@ericsson.com

Nazin Hossain
Ericsson
Decarie Boulevard
Town of Mount Royal, Quebec
Canada

Email: Nazin.Hossain@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2014

P. Saint-Andre
Cisco Systems, Inc.
A. Houri
IBM
J. Hildebrand
Cisco Systems, Inc.
July 1, 2013

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

draft-ietf-stox-core-00

Abstract

As a foundation for the definition of bidirectional protocol mappings between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP), this document specifies the architectural assumptions underlying such mappings as well as the mapping of addresses and error conditions.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
3. Architectural Assumptions 3
4. Address Mapping 5
4.1. Overview 5
4.2. Local Part Mapping 6
4.3. Instance-Specific Mapping 7
4.4. SIP to XMPP 8
4.5. XMPP to SIP 8
5. Error Condition Mapping 9
5.1. XMPP to SIP 10
5.2. SIP to XMPP 10
6. Security Considerations 12
7. IANA Considerations 12
8. References 12
8.1. Normative References 12
8.2. Informative References 13
Appendix A. Acknowledgements 13
Authors' Addresses 14

1. Introduction

The IETF has worked on two signalling technologies that can be used for multimedia session negotiation, messaging, presence, capabilities discovery, notifications, and other application-level functionality:

- The Session Initiation Protocol [RFC3261], along with various SIP extensions developed within the SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) Working Group.
- The Extensible Messaging and Presence Protocol [RFC6120], along with various XMPP extensions developed by the IETF as well as by the XMPP Standards Foundation.

Because these technologies are widely deployed, it is important to clearly define mappings between them for the sake of interworking. This document inaugurates a series of SIP-XMPP interworking specifications by defining the architectural assumptions underlying such mappings as well as the mapping of addresses and error conditions.

The discussion venue for this document is the mailing list of the STOX WG; visit https://www.ietf.org/mailman/listinfo/stox for subscription information and discussion archives.

2. Terminology

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].

3. Architectural Assumptions

Protocol translation between SIP and XMPP could occur in a number of different entities, depending on the architecture of real-time communication deployments. For example, protocol translation could occur within a multi-protocol server (which uses application-specific connection managers to initiate traffic to and accept traffic from clients or other servers natively using SIP/SIMPLE, XMPP, etc.), within a multi-protocol client (which enables a user to establish connections natively with various servers using SIP/SIMPLE, XMPP, etc.), or within a gateway that acts as a dedicated protocol translator (which takes one protocol as input and provides another protocol as output).

This document assumes that the protocol translation will occur within

a gateway. (This assumption not meant to discourage protocol translation within multi-protocol clients or servers; instead, this assumption is followed mainly to clarify the discussion and examples so that the protocol translation principles can be more easily understood and can be applied by client and server implementors with appropriate modifications to the examples and terminology.) Specifically, we assume that the protocol translation will occur within an "XMPP-to-SIP gateway" that translates XMPP syntax and semantics on behalf of an XMPP service when communicating with SIP services and/or within a "SIP-to-XMPP gateway" that translates SIP syntax and semantics on behalf of a SIP service when communicating with XMPP services (naturally, these logical functions could occur in one and the same actual translator).

This document assumes that a gateway will translate directly from one protocol to the other. For the sake of the examples, we further assume that protocol translation will occur within a gateway in the source domain, so that information generated by the user of an XMPP service will be translated by a gateway within the trust domain of that XMPP service, and information generated by the user of a SIP service will be translated by a gateway within the trust domain of that SIP service. However, nothing in this document ought to be taken as recommending against protocol translation at the destination domain.

An architectural diagram for a possible gateway deployment is shown below, where the entities have the following significance and the "#" character is used to show the boundary of a trust domain:

- o romeo@example.net -- a SIP user.
- o example.net -- a SIP service with a gateway ("GW") to XMPP.
- o juliet@example.com -- an XMPP user.
- o example.com -- an XMPP service with a gateway ("GW") to SIP.

```
##@#########################################################
#                           #                             #
#    +-------------+----+   #   +----+-------------+     #
#    | example.net | GW |---#---| GW | example.com |     #
#    +-------------+----+   #   +----+-------------+     #
#          |                #              |             #
#     romeo@example.net     #        juliet@example.com  #
#                           #                             #
#############################################################
```

4. Address Mapping

4.1. Overview

The basic SIP address format is a "sip:" or "sips:" URI as specified in [RFC3261]. When a SIP entity supports extensions for instant messaging it might be identified by an 'im:' URI as specified in the Common Profile for Instant Messaging [RFC3860] (see [RFC3428]) and when a SIP entity spports extensions for presence it might be identified by a 'pres:' URI as specified in the Common Profile for Presence [RFC3859] (see [RFC3856]).

The XMPP address format is specified in [RFC6122]; as discussed in [RFC6121], instant messaging and presence applications of XMPP also need to support 'im:' and 'pres:' URIs as specified in [RFC3860] and [RFC3859] respectively, although such support might simply involve leaving resolution of such addresses up to an XMPP server.

In this document we primarily describe mappings for addresses of the form <user@domain>; however, we also provide guidelines for mapping the addresses of specific user agent instances, which take the form of Globally Routable User Agent URIs (GRUUs) in SIP and "resourceparts" in XMPP. Mapping of protocol-specific identifiers (such as telephone numbers) is out of scope for this specification. In addition, we have ruled the mapping of domain names as out of scope for now since that is a matter for the Domain Name System; specifically, the issue for interworking between SIP and XMPP relates to the translation of fully internationalized domain names (IDNs) into non-internationalized domain names (IDNs are not allowed in the SIP address format, but are allowed in the XMPP address via Internationalized Domain Names in Applications, see [RFC6122] and [I-D.ietf-xmpp-6122bis]). Therefore, in the following sections we focus primarily on the local part of an address (these are called variously "usernames", "instant inboxes", "presentities", and "localparts" in the protocols at issue), secondarily on the instance-specific part of an address, and not at all on the domain-name part of an address.

The sip:/sips:, im:/pres:, and XMPP address schemes allow different sets of characters (although all three allow alphanumeric characters and disallow both spaces and control characters). In some cases, characters allowed in one scheme are disallowed in others; these characters need to be mapped appropriately in order to ensure interworking across systems.

4.2. Local Part Mapping

The local part of a sip:/sips: URI inherits from the "userinfo" rule in [RFC3986] with several changes; here we discuss the SIP "user" rule only:

```
user             =  1*( unreserved / escaped / user-unreserved )
user-unreserved  =  "&" / "=" / "+" / "$" / "," / ";" / "?" / "/"
unreserved       =  alphanum / mark
mark             =  "-" / "_" / "." / "!" / "~" / "*" / "'"
                    / "(" / ")"
```

Here we make the simplifying assumption that the local part of an im:/pres: URI inherits from the "dot-atom-text" rule in [RFC5322] rather than the more complicated "local-part" rule:

```
dot-atom-text =  1*atext *("." 1*atext)
atext         =  ALPHA / DIGIT /    ; Any character except
                 "!" / "#" / "$" /  ; controls, SP, and
                 "%" / "&" / "'" /  ; specials. Used for
                 "*" / "+" / "-" /  ; atoms.
                 "/" / "=" / "?" /
                 "^" / "_" / "`" /
                 "{" / "|" / "}" /
                 "~"
```

The local part of an XMPP address allows any ASCII character except space, controls, and the " & ' / : < > @ characters.

To summarize the foregoing information, the following table lists the allowed and disallowed characters in the local part of identifiers for each protocol (aside from the alphanumeric, space, and control characters), in order by hexadecimal character number (where each "A" row shows the allowed characters and each "D" row shows the disallowed characters).

Table 1: Allowed and disallowed characters

```
+---+-----------------------------------+
| SIP/SIPS CHARACTERS                   |
+---+-----------------------------------+
| A | !  $ &'()*+,-./ ; = ?     _     ~ |
| D |  "# %          : < > @[\]^ '{|}  |
+---+-----------------------------------+
| IM/PRES CHARACTERS                    |
+---+-----------------------------------+
| A | ! #$%&'  *+ - /   = ?    ^_'{|}~ |
| D |  "     ()  , . :;< > @[\]        |
+---+-----------------------------------+
| XMPP CHARACTERS                       |
+---+-----------------------------------+
| A | ! #$%  ()*+,-.  ; = ? [\]^_'{|}~ |
| D |  "   &'       /: < > @           |
+---+-----------------------------------+
```

When transforming the local part of an address from one scheme to another, an application SHOULD proceed as follows:

1. Unescape any escaped characters in the source address (e.g., from SIP to XMPP unescape "%2F" to "/" and from XMPP to SIP unescape "\27" to "'").
2. Leave unmodified any characters that are allowed in the destination scheme.
3. Escape any characters that are allowed in the source scheme but reserved in the destination scheme, as escaping is defined for the destination scheme. In particular:
   * Where the destination scheme is a URI (i.e., an im:, pres:, sip:, or sips: URI), each reserved character MUST be percent-encoded to "%hexhex" as specified in Section 2.6 of [RFC4395] (e.g., when transforming from XMPP to SIP, encode "/" as "%2F").
   * Where the destination scheme is a native XMPP address, each reserved character MUST be encoded to "\hexhex" as specified in [XEP-0106] (e.g., when transforming from SIP to XMPP, encode "'" as "\27").

4.3. Instance-Specific Mapping

The meaning of a resourcepart in XMPP (i.e., the portion of a JID after the slash character, such as "foo" in "user@example.com/foo") matches that of a Globally Routable User Agent URI (GRUU) in SIP [RFC5627]. In both cases, these constructs identify a particular device associated with the bare JID ("localpart@domainpart") of an XMPP entity or with the Address of Record (AOR) of a SIP entity.

Therefore, it is reasonable to map the value of a "gr" URI parameter to an XMPP resource part, and vice-versa.

Note that the "gr" URI parameter in SIP can contain only characters from the ASCII range, whereas an XMPP resourcepart can contain nearly any Unicode character [UNICODE]. Therefore Unicode characters outside the ASCII range need to be mapped to characters in the ASCII range, as described below.

4.4. SIP to XMPP

The following is a high-level algorithm for mapping a sip:, sips:, im:, or pres: URI to an XMPP address:

1. Remove URI scheme.
2. Split at the first '@' character into local part and hostname (mapping the latter is out of scope).
3. Translate any percent-encoded strings ("%hexhex") to percent-decoded octets.
4. Treat result as a UTF-8 string.
5. Translate "&" to "\26", "'" to "\27", and "/" to "\2f" respectively in order to properly handle the characters disallowed in XMPP addresses but allowed in sip:/sips: URIs and im:/pres: URIs as shown in Table 1 above (this is consistent with [XEP-0106]).
6. Apply Nodeprep profile of Stringprep [RFC3454] or its replacement (see [RFC6122] and [I-D.ietf-xmpp-6122bis]) for canonicalization (OPTIONAL).
7. Recombine local part with mapped hostname to form a bare JID ("localpart@domainpart").
8. If the (SIP) address contained a "gr" URI parameter, append a slash character "/" and the "gr" value to the bare JID to form a full JID ("localpart@domainpart/resourcepart").

4.5. XMPP to SIP

The following is a high-level algorithm for mapping an XMPP address to a sip:, sips:, im:, or pres: URI:

1. Split XMPP address into localpart (mapping described in remaining steps), domainpart (hostname; mapping is out of scope), and resourcepart (specifier for particular device or connection, for which an OPTIONAL mapping is described below).
2. Apply Nodeprep profile of [RFC3454] or its replacement (see [RFC6122] and [I-D.ietf-xmpp-6122bis]) for canonicalization of the XMPP localpart (OPTIONAL).

3. Translate "\26" to "&", "\27" to "'", and "\2f" to "/" respectively (this is consistent with [XEP-0106]).
4. Determine if the foreign domain supports im: and pres: URIs (discovered via [RFC2782] lookup as specified in [RFC6121]), else assume that the foreign domain supports sip:/sips: URIs.
5. If converting into im: or pres: URI, for each byte, if the byte is in the set (),.;[\] or is a UTF-8 character outside the ASCII range then percent-encode that byte to "%hexhex" format. If converting into sip: or sips: URI, for each byte, if the byte is in the set #%[\]^'{|} or is a UTF-8 character outside the ASCII range then percent-encode that byte to "%hexhex" format.
6. Combine resulting local part with mapped hostname to form local@domain address.
7. Prepend with 'im:' scheme (for XMPP <message/> stanzas) or 'pres:' scheme (for XMPP <presence/> stanzas) if foreign domain supports these, else prepend with 'sip:' or 'sips:' scheme according to local service policy.
8. If the XMPP address included a resourcepart and the destination URI scheme is 'sip:' or 'sips:', optionally append the slash character '/' and then append the resourcepart (making sure to percent-encode any UTF-8 characters outside the ASCII range) as the "gr" URI parameter.

5. Error Condition Mapping

SIP response codes are specified in [RFC3261] and XMPP error conditions are specified in [RFC6120]. Because there is no equivalent in XMPP for the provisional (1xx) and successful (2xx) response codes in SIP, mappings are provided only for the redirection (3xx), request failure (4xx), server failure (5xx), and glogal failure (6xx) codes.

5.1. XMPP to SIP

Table 8: Mapping of XMPP error conditions to SIP response codes

XMPP Error Condition	SIP Response Code
<bad-request/>	400
<conflict/>	400
<feature-not-implemented/>	501
<forbidden/>	403
<gone/>	410
<internal-server-error/>	500
<item-not-found/>	404
<jid-malformed/>	484
<not-acceptable/>	406
<not-allowed/>	405
<not-authorized/>	401
<recipient-unavailable/>	480
<redirect/>	300
<registration-required/>	407
<remote-server-not-found/>	502
<remote-server-timeout/>	504
<resource-constraint/>	500
<service-unavailable/>	503
<subscription-required/>	407
<undefined-condition/>	400
<unexpected-request/>	491

5.2. SIP to XMPP

The mapping of SIP response codes to XMPP error conditions SHOULD be as follows (note that XMPP does not include 100-series or 200-series response codes, only error conditions):

Table 9: Mapping of SIP response codes to XMPP error conditions

SIP Response Code	XMPP Error Condition
300	<redirect/>
301	<gone/>
302	<redirect/>
305	<redirect/>
380	<not-acceptable/>
400	<bad-request/>
401	<not-authorized/>
402	see note [1]
403	<forbidden/>
404	<item-not-found/>
405	<not-allowed/>
406	<not-acceptable/>
407	<registration-required/>
408	<recipient-unavailable/>
410	<gone/>
413	<bad-request/>
414	<bad-request/>
415	<bad-request/>
416	<bad-request/>
420	<bad-request/>
421	<bad-request/>
423	<bad-request/>
480	<recipient-unavailable/>
481	<item-not-found/>
482	<not-acceptable/>
483	<not-acceptable/>
484	<jid-malformed/>
485	<item-not-found/>
486	<recipient-unavailable/>
487	<recipient-unavailable/>
488	<not-acceptable/>
491	<unexpected-request/>
493	<bad-request/>
500	<internal-server-error/>
501	<feature-not-implemented/>
502	<remote-server-not-found/>
503	<service-unavailable/>
504	<remote-server-timeout/>
505	<not-acceptable/>
513	<bad-request/>
600	<recipient-unavailable/>
603	<recipient-unavailable/>
604	<item-not-found/>
606	<not-acceptable/>

1. The XMPP <payment-required/> error condition was removed in [RFC6120].

6. Security Considerations

Detailed security considerations for SIP are given in [RFC3261] and for XMPP in [RFC6120].

7. IANA Considerations

This document requests no actions of IANA.

8. References

8.1. Normative References

[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.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", RFC 4395, February 2006.

[RFC5627] Rosenberg, J., "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)", RFC 5627, October 2009.

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

[RFC6122] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 6122, March 2011.

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 6.2", 2012, <http://www.unicode.org/versions/Unicode6.2.0/>.

8.2. Informative References

[I-D.ietf-xmpp-6122bis]
Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", draft-ietf-xmpp-6122bis-07 (work in progress), April 2013.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("STRINGPREP")", RFC 3454, December 2002.

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

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

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

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

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

[XEP-0106]
Saint-Andre, P. and J. Hildebrand, "JID Escaping", XSF XEP 0106, May 2005.

Appendix A. Acknowledgements

The authors wish to thank the following individuals for their feedback: Fabio Forno, Adrian Georgescu, Saul Ibarra, Markus Isomaki, Salvatore Loreto, Daniel-Constantin Mierla, Tory Patnoe, and Robert Sparks.

Authors' Addresses

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3282
Email: psaintan@cisco.com

Avshalom Houri
IBM
Building 18/D, Kiryat Weizmann Science Park
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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2014

P. Saint-Andre
Cisco Systems, Inc.
S. Ibarra
AG Projects
S. Loreto
Ericsson

July 1, 2013

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

draft-ietf-stox-groupchat-00

Abstract

This document defines a bidirectional protocol mapping for the exchange of instant messages in the context of a multiparty chat session among users of the Session Initiation Protocol (SIP) and users of the Extensible Messaging and Presence Protocol (XMPP). Specifically, this document defines a mapping between the SIP-based Message Session Relay Protocol (MSRP) and the XMPP Multi-User Chat (MUC) extension.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
3. XMPP MUC to MSRP Multi-party Messaging Session 3
3.1. Enter Room 5
3.2. Set Nickname 8
3.3. Change Nickname 10
3.4. Invite Another User to a Room 10
3.5. Presence Broadcast 12
3.6. Exchange Messages 15
3.6.1. Send a Message to All Occupants 15
3.6.2. Send a Private Message 16
3.7. Exit Room 18
4. MSRP Multi-party Messaging Session to XMPP MUC 19
4.1. Enter Room 20
4.2. Change Nickname 22
4.3. Invite Another User to a Room 23
4.4. Presence Broadcast 23
4.5. Exchange Messages 25
4.5.1. Send a Message to All Occupants 26
4.5.2. Send a Private Message 27
4.6. Exit Room 28
5. Handling of Nicknames and Display Names 29
6. Security Considerations 29
7. IANA Considerations 30
8. References 30
8.1. Normative References 30
8.2. Informative References 31
Appendix A. Acknowledgements 31
Authors' Addresses 32

1. Introduction

Both the Session Initiation Protocol (SIP) [RFC3261] and the Extensible Messaging and Presence Protocol (XMPP) [RFC6120] can be used for the purpose of multiparty text chat over the Internet. To ensure interworking between these technologies, it is important to define bidirectional protocol mappings.

The architectural assumptions underlying such protocol mappings are provided in [I-D.ietf-stox-core], including mapping of addresses and error conditions. This document specifies mappings for multiparty text chat sessions (often called "groupchat"); specifically, this document defines a mapping between the XMPP Multi-User Chat (MUC) extension [XEP-0045] and SIP-based multiparty chat using Message Session Relay Protocol [RFC4975] as specified in [I-D.ietf-simple-chat].

Both MUC and MSRP contain a large set of features, such as the ability to administer rooms, kick and ban users, reserve a nickname within a room, change room subject, enable room moderation, and destroy the room. This document covers only a basic subset of groupchat features: joining the room, establishing or changing a room nickname, inviting another user to the room, modifying presence information within the room, sending a message to all participants, sending a private message to a single participant, and leaving the room. Future documents might define mappings for additional features beyond this set.

The discussion venue for this document is the mailing list of the STOX WG; visit https://www.ietf.org/mailman/listinfo/stox for subscription information and discussion archives.

2. Terminology

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].

A number of technical terms used here are defined in [RFC3261], [RFC4975], [RFC6120], and [XEP-0045]. The term "JID" is short for "Jabber Identifier".

3. XMPP MUC to MSRP Multi-party Messaging Session

This section describes how to map an XMPP MUC session to an MSRP

Multi-party Messaging session. The following diagram outlines the overall protocol flow.

```
XMPP User                    Gateway               MSRP Conference
    |                           |                          |
    |(F1) (XMPP) Enter room     |                          |
    |-------------------------->|                          |
    |                           |(F2) (SIP) INVITE         |
    |                           |------------------------->|
    |                           |(F3) (SIP) 200 OK         |
    |                           |<-------------------------|
    |                           |(F4) (SIP) ACK            |
    |                           |------------------------->|
    |                           |(F5) (MSRP) NICKNAME      |
    |                           |------------------------->|
    |                           |(F6) (MSRP) 200 OK        |
    |                           |<-------------------------|
    |                           |(F7) (SIP)SUBSCRIBE       |
    |                           |------------------------->|
    |                           |     Event:conference     |
    |                           |                          |
    |                           |(F8) (SIP) 200 OK         |
    |                           |<-------------------------|
    |                           |(F9) (SIP) NOTIFY         |
    |                           |<-------------------------|
    |                           |(F10) (SIP) 200 OK        |
    |                           |------------------------->|
    |(F11) (XMPP) Presence      |                          |
    |<--------------------------|                          |
    |(F12) (XMPP) Subject       |                          |
    |<--------------------------|                          |
    |                           |                          |
    |(F13) (XMPP) Chat message  |                          |
    |-------------------------->|                          |
    |                           |(F14) (MSRP) SEND         |
    |                           |------------------------->|
    |                           |(F15) (MSRP) 200 OK       |
    |                           |<-------------------------|
    |                           |                          |
    |(F16) (XMPP) Chat message  |                          |
    |<--------------------------|                          |
    .                           .                          .
    .                           .                          .
    |(F17) (XMPP) Exit room     |                          |
    |-------------------------->|                          |
    |                           |(F18) (SIP) BYE           |
    |                           |------------------------->|
    |                           |(F19) (SIP) 200 OK        |
```

```
    |                          |<-------------------------|
```

Detailed protocol flows and mappings are provided in the following sections.

3.1. Enter Room

As defined in the XMPP Multi-User Chat (MUC) extensions [XEP-0045], when an XMPP user (say, juliet@example.com) wants to join a groupchat room (say, "verona@chat.example.org"), she sends a <presence/> stanza to that chat room. In her request she also specifies the nickname she wants to use within the room (say, "JuliC"); in XMPP this Room Nickname is the resourcepart of an Occupant JID (thus "verona@chat.example.org/JuliC"). The joining client signals its ability to speak the multi-user chat protocol by including in the initial presence stanza an empty <x/> element qualified by the 'http://jabber.org/protocol/muc' namespace.

Example: (F1) Juliet enters room

```
<presence from='juliet@example.com/balcony'
          to='verona@chat.example.org/JuliC'>
  <x xmlns='http='http://jabber.org/protocol/muc'/>
</presence>
```

Upon receiving such a presence stanza, the XMPP server to which Juliet has authenticated attempts to (a) deliver the stanza to a local domain or (b) route the presence stanza to the remote domain that services the hostname in the 'to' attribute. In this document we assume that the hostname in the 'to' attribute is a groupchat-aware SIP/MSRP service hosted by a separate server.

As specified in [RFC6121], the XMPP server needs to determine the identity of the remote domain, which it does by performing one or more DNS SRV lookups [RFC2782]. For presence stanzas, the order of lookups recommended by [RFC6121] is to first try the "_xmpp-server" service as specified in [RFC6120] and to then try the "_pres" service as specified in [RFC3861]. Here we assume that the first lookup will fail but that the second lookup will succeed and return a resolution "_pres._s2x.example.org", since we have already assumed that the example.org hostname is running a SIP instant messaging service. (Note: The XMPP server might have previously determined that the remote domain is a SIMPLE server, in which case it would not need to perform the SRV lookups; the caching of such information is a matter of implementation and local service policy, and is therefore out of scope for this document.)

Once the XMPP server (example.com) has determined that the remote

domain is serviced by a SIMPLE server, it hands the XMPP presence stanza off to its local XMPP-to-SIP gateway code (this might be a specialized connection manager within the example.com service or might be a dedicated component at, say, x2s.example.com), which transforms the presence stanza into SIP syntax and routes it to the remote conference server (chat.example.org).

Because a multi-user chat service accepts the presence stanza shown above as a request to enter a room, the XMPP-to-SIP gateway transforms it in a SIP INVITE request.

Example: (F2) Juliet enters room (SIP conversion)

```
INVITE sip:verona@chat.example.org SIP/2.0
To: <sip:verona@chat.example.org>
From: "Juliet" <sip:juliet@example.com>
Contact: <sip:juliet@example.com>;gr=balcony
Call-ID: 711609sa
Content-Type: application/sdp
Content-Length: [length]

c=IN IP4 x2s.example.org
m=message 7654 TCP/MSRP *
a=accept-types:text/cpim
a=accept-wrapped-types:text/plain text/html
a=path:msrp://x2s.example.com:7654/jshA7weztas;tcp
a=chatroom
```

Here the Session Description Protocol offer specifies the MSRP-aware XMPP-to-SIP gateway on the XMPP side as well as other particulars of the session.

> There is no direct mapping for the MSRP URIs. In fact MSRP URIs identify a session of instant messages at a particular device; they are ephemeral and have no meaning outside the scope of that session. The authority component of the MSRP URI MUST contain the XMPP-to-SIP gateway hostname or numeric IP address and an explicit port number.

As specified in [I-D.ietf-stox-core], the mapping of XMPP syntax elements to SIP and [RFC4566] syntax elements is as shown in the following table.

Table 1: Message syntax mapping from XMPP to SIP/SDP

XMPP Element or Attribute	SIP Header or SDP Contents
from	From
to (without the /nick)	To

Here we assume that the MSRP conference server accepts the session establishment. It includes the 'isfocus' and other relevant feature tags in the Contact header field of the response. The MSRP confernece server also includes an answer session description that acknowledges the choice of media and contains the extensions specified in [I-D.ietf-simple-chat].

Example: (F3) Chat room accepts session establishment

```
SIP/2.0 200 OK
From: <sip:verona@chat.example.org>
To: "Juliet" <sip:juliet@example.com>;tag=786
Call-ID: 711609sa
Contact: <sip:verona@chat.example.org;transport=tcp>\
         ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL,SUBSCRIBE,NOTIFY"\
         ;automata;isfocus;message;event="conference"
Content-Type: application/sdp
Content-Length: [length]

c=IN IP4 example.org
m=message 12763 TCP/MSRP *
a=chatroom:nickname private-messages
a=accept-types:message/cpim
a=accept-wrapped-types:text/plain text/html *
a=path:msrp://s2x.example.org:12763/kjhd37s2s20w2a;tcp
```

Upon receiving such a response, the SIMPLE server or associated SIP-to-XMPP gateway sends a SIP ACK to the MSRP conference server on behalf of the joining user.

Example: (F4) Gateway sends ACK to MSRP conference server

```
ACK sip:verona@chat.example.org SIP/2.0
To: <sip:verona@chat.example.org>;tag=087js
From: "Juliet" <sip:juliet@example.com>;tag=786
Call-ID: 711609sa
```

3.2. Set Nickname

If the chat room server accepted the session, the SIMPLE server or associated SIP-to-XMPP gateway MUST set up the nickname as received in the presence stanza (i.e., the resourcepart of the 'to' address, such "JuliC" in "verona@chat.example.org/JuliC"). The nickname is set up using the extension specified in [I-D.ietf-simple-chat].

Example: (F5) Gateway sets up nickname

```
MSRP a786hjs2 NICKNAME
To-Path: msrp://s2x.example.org:12763/kjhd37s2s20w2a;tcp
From-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
Use-Nickname: "JuliC"
-------a786hjs2
```

The MSRP conference server analyzes the existing allocation of nicknames, accepts the nickname proposal and answers with a 200 response.

Example: (F6) MSRP conference accepts nickname proposal

```
MSRP a786hjs2 200 OK
To-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
From-Path: msrp://s2x.example.org:12763/kjhd37s2s20w2a;tcp
-------a786hjs2
```

So far we have assumed that the requested nickname did not conflict with any existing nicknames. The following text describes the handling of a nickname conflict.

```
                                                      MSRP conference
   XMPP User                   Gateway                      server
       |                          |                          |
       |(F1) (XMPP) Enter room    |                          |
       |------------------------->|                          |
       |                          |(F2) (SIP) INVITE         |
       |                          |------------------------->|
       |                          |(F3) (SIP) 200 OK         |
       |                          |<-------------------------|
       |                          |(F4) (SIP) ACK            |
       |                          |------------------------->|
       |                          |(F5) (MSRP) NICKNAME      |
       |                          |------------------------->|
       |                          |(F6) (MSRP) 425 Error     |
       |                          |<-------------------------|
       |                          |                          |
       |(F7) (XMPP) Presence Error
       |<-------------------------|                          |
       .                          .                          .
       |                          |(F8) (SIP) BYE            |
       |                          |------------------------->|
       |                          |(F9) (SIP) 200 OK         |
       |                          |<-------------------------|
```

The MSRP conference server analyzes the existing allocation of nicknames, and detects that the nickname proposal is already provided to another participant. In this case the MSRP conference server answers with a 425 response.

Example: (F6) MSRP conference does not accept nickname proposal

```
MSRP a786hjs2 425 Nickname usage failed
To-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
From-Path: msrp://s2x.example.org:12763/kjhd37s2s20w2a;tcp
-------a786hjs2
```

Upon receiving such a response, the SIP-to-XMPP gateway SHOULD translate it into an XMPP presence stanza of type "error" specifying a <conflict/> error condition (which implies that the XMPP client will then need to choose another nickname and repeat the process of joining).

Example: (F7) Conflict error for nickname

```
<presence from='verona@chat.example.org'
          to='juliet@example.com/balcony'
          type='error'>
  <x xmlns='http='http://jabber.org/protocol/muc'/>
  <error type='cancel'>
    <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</presence>
```

Alternatively, the gateway might generate a new nickname request on behalf of the XMPP user, thus shielding the XMPP client from handling the conflict error.

3.3. Change Nickname

The XMPP user might want to change her nickname. She can do so by sending an updated presence stanza to the room, containing a new nickname.

```
                                                   MSRP conference
XMPP User                   Gateway                      server
    |                          |                          |
    |(F1) (XMPP) Presence to change Nickname              |
    |------------------------->|                          |
    |                          |(F2) (MSRP) NICKNAME      |
    |                          |------------------------->|
    |                          |(F3) (MSRP) 200 OK        |
    |                          |<-------------------------|
```

Example: (F1) Juliet changes her nickname

```
<presence from='juliet@example.com/balcony'
          to='verona@chat.example.org/Juliet'/>
```

The nickname change is handled as described above.

3.4. Invite Another User to a Room

In XMPP there are two methods for inviting another user to a room: direct invitations [XEP-0249] (sent directly from the user's real JID outside the room to the invitee's real JID) and mediated invitations (sent through the room from the user's Occupant JID to the invitee's JID). In this document we cover mediated invitations only.

```
XMPP User                    Gateway               MSRP Conference
    |                           |                         |
    |(F1) (XMPP) Message stanza to invite participant     |
    |-------------------------->|                         |
    |                           |(F2) (SIP) REFER         |
    |                           |------------------------->|
    |                           |(F3) (SIP) 200 OK        |
    |                           |<-------------------------|
    .                           .                         .
    |                           |(F4) (SIP) NOTIFY        |
    |                           |<-------------------------|
```

For example, if Juliet decides to invite Benvolio to the room, she sends a message stanza with an invite and Benvolio's JID (which could be his real JID or an Occupant JID in another room).

Example: (F1) Juliet invites Hecate to the room

```
<message from='juliet@example.com/balcony'
         id='nzd143v8'
         to='verona@chat.example.org'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <invite to='benvolio@example.com'/>
  </x>
</message>
```

The SIP - XMPP gateway then sends a SIP REFER request to the MSRP conference server indicating who needs to be invited in the Refer-To header, as per [RFC4579] (sec 5.5)

Example: (F2) SIP translation of invite

```
REFER sip:verona@chat.example.com SIP/2.0
Via: SIP/2.0/UDP client.example.com;branch=z9hG4bKg4534
Max-Forwards: 70
To: <sip:verona@chat.example.com>
From: "Juliet" <sip:juliet@example.com>;tag=5534562
Call-ID: 849392fklgl43
CSeq: 476 REFER
Contact: <sip:juliet@juliet.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Accept: message/sipfrag
Refer-To: <sip:benvolio@example.com>
Supported: replaces
Content-Length: 0
```

The progress of the invitation will be tracked by the reeived NOTIFY

requests as per [RFC3515].

Example: (F4) Progress notification for invitation

```
NOTIFY sip:juliet@example.com SIP/2.0
Via: SIP/2.0/UDP client.example.com;branch=z9hG4bK9922ef992-25
To: <sip:juliet@example.com>;tag=5534562
From: <sip:verona@chat.example.com>;tag=18747389
Call-ID: 849392fklgl43
CSeq: 1993402 NOTIFY
Max-Forwards: 70
Event: refer
Subscription-State: active;expires=60
Contact: sip:verona@chat.example.com
Content-Type: message/sipfrag;version=2.0
Content-Length: ...

SIP/2.0 200 OK
```

3.5. Presence Broadcast

If the MSRP conference service accepts the request to enter a room, the XMPP user expects to receive back presence information from all the existing occupants of the room. So the XMPP-to-SIP gateway MUST subscribe to the Conference Event package [RFC4575] on the MSRP conference server. When the subscription is completed the MSRP conference server sends to the XMPP-to-SIP gateway a NOTIFY containing the presence information of all the existing occupants, represented using the [RFC4575] format.

Example: (F9) MSRP conference sends presence information

```
NOTIFY sip:verona@chat.example.org SIP/2.0
To: "Juliet" <sip:juliet@example.com>;gr=balcony
From: <sip:verona@chat.example.org>;tag=a3343df32
Call-ID: k3l43id034ksereree
Event: conference
Subscription-State: active;expires=3600
Content-Type: application/conference-info+xml
Content-Length: ...

<conference-info version="0" state="full"
    entity="sip:3402934234@chat.example.org">
  <conference-description>
    <subject>Today in Verona</subject>
    <conf-uris>
      <entry>
        <uri>tel:+18882934234</uri>
      </entry>
    </conf-uris>
  </conference-description>
  <users>
    <user entity="sip:verona@chat.example.org;gr=Romeo"
          state="full">
      <display-text>Romeo</display-text>
      <roles>
        <entry>participant</entry>
      </roles>
    </user>
    <user entity="sip:verona@chat.example.org;gr=Ben"
          state="full">
      <display-text>Ben</display-text>
      <roles>
        <entry>participant</entry>
      </roles>
    </user>
  </users>
</conference-info>
```

The following table shows the syntax mapping from the RFC 4575 payload to the XMPP participants list. (Mappings for elements not mentioned are undefined.)

Table 2: Participant list mapping

RFC 4575 Element	XMPP Element or Attribute
conference-info entity	room JID
conference subject	room subject
user entity	participant bare JID
user display-text / nickname	participant nickname
endpoint entity	participant full JID

Upon receiving such a response, the SIP-to-XMPP gateway MUST send a 200 OK to the MSRP conference server and translate the participant list into a series of XMPP presence stanzas.

Example: (F11) Chatroom presence information translated into XMPP

```
<presence from='verona@chat.example.org/Romeo'
          to='verona@chat.example.org/JuliC'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='none' role='participant'/>
  </x>
</presence>

<presence from='verona@chat.example.org/Ben'
          to='verona@chat.example.org/JuliC'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='none' role='participant'/>
  </x>
</presence>
```

If the NOTIFY included a subject, the gateway SHALL convert it into a separate XMPP message.

Example: (F12) Chatroom subject translated into XMPP

```
<message from='verona@chat.example.com/mayor'
         to='juliet@example.com/balcony'
         id='mbh2vd68'>
  <subject>Today in Verona</subject>
</message>
```

The mapping of SIP and [RFC4575] payload syntax elements to XMPP syntax elements is as shown in the following table. (Mappings for elements not mentioned are undefined.)

Table 2: Message syntax mapping from SIP to XMPP

SIP Header or RFC4575 Contents	XMPP Element or Attribute
<user entity=...>	From
To + / <display-text>	To
roles	role
'none'	affiliation

3.6. Exchange Messages

Once the user has joined the chatroom, the user can exchange an unbounded number of messages both public and private.

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

Table 3: Message syntax mapping from XMPP Message to MSRP

XMPP Element or Attribute	CPIM Header
to	To
from	From
<body/>	body of the SEND request

3.6.1. Send a Message to All Occupants

When Juliet wants to sends a message to all other occupants in the room, she sends a message of type "groupchat" to the <room@service> itself (in our example, <verona@chat.example.org>).

The following examples show an exchange of a public message.

Example: (F13) Juliet sends message to all occupants

```
<message from='juliet@example.com/balcony'
         to='verona@chat.example.org'
         type='groupchat'
         id='lzfed24s'>
      <body>Who knows where Romeo is?</body>
</message>
```

Upon receiving such a message, the XMPP-to-SIP gateway MUST translate

it into an MSRP SEND message.

Example: (F14) Gateway transforms XMPP message to MSRP

```
MSRP a786hjs2 SEND
To-Path: msrp://s2x.example.org:12763/kjhd37s2s20w2a;tcp
From-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
Message-ID: 87652491
Byte-Range: 1-*/*
Content-Type: message/cpim

To: <sip:verona@chat.example.org>
From: "Juliet" <sip:juliet@example.com>
DateTime: 2008-10-15T15:02:31-03:00
Content-Type: text/plain

Who knows where Romeo is?
-------a786hjs2$
```

Upon receiving the SEND request, if the request either contains a Failure-Report header field value of "yes" or does not contain a Failure-Report header at all, the MSRP conference server MUST immediately generate and send a response.

```
MSRP d93kswow 200 OK
To-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
From-Path: msrp://s2x.example.org:12763/kjhd37s2s20w2a;tcp
-------d93kswow$
```

Since an XMPP MUC room could be moderated and an XMPP user cannot be sure whether her message has been accepted or not without receiving it back from the server, [XEP-0045] states that the sender needs to receive the same message it has generated. So in this scenario the XMPP-to-SIP gateway has to reflect the message back to the sender. This prodedure only applies to XMPP endpoints.

3.6.2. Send a Private Message

Since each occupant has a unique JID, Juliet can send a "private message" to a selected occupant through the service by sending a message to the user’s occupant JID. The XMPP message type SHOULD be "chat" and MUST NOT be "groupchat", but MAY be left unspecified.

If the XMPP-to-SIP gateway has support for private messaging it MUST advertise that fact by adding a "private-messages" value to the a=chatroom SDP attribute it sends to the MSRP conference server, as specified in [I-D.ietf-simple-chat].

```
a=chatroom:nickname private-messages
```

The following examples show an exchange of a private message.

Example: Juliet sends private message

```
<message from='juliet@example.com/balcony'
         to='verona@chat.example.org/Romeo'
         type='chat'
         id='6sfln45q'/>
      <body>O Romeo, Romeo! wherefore art thou Romeo?</body>
</message>
```

Upon receiving such a message, the XMPP-to-SIP gateway MUST translate it into an MSRP SEND message.

Example: Gateway transforms private message from XMPP to MSRP

```
MSRP a786hjs2 SEND
To-Path: msrp://s2x.example.org:12763/kjhd37s2s20w2a;tcp
From-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
Message-ID: 87652491
Byte-Range: 1-*/*
Content-Type: message/cpim

To: <sip:verona@chat.example.org>;gr=Romeo
From: <sip:juliet@example.org>;gr=balcony
DateTime: 2008-10-15T15:02:31-03:00
Content-Type: text/plain

O Romeo, Romeo! wherefore art thou Romeo?
-------a786hjs2$
```

The MSRP conference server is responsible for sending the message to the intended recipient, and when doing so MUST modify the "From" header to the sender's address within the chatroom.

Example: MSRP conference sends private message to SIP user

```
MSRP a786hjs2 SEND
To-Path: msrp://s2x.example.org:12763/kjhd37s2s20w2a;tcp
From-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
Message-ID: 87652491
Byte-Range: 1-*/*
Content-Type: message/cpim

To: <sip:romeo@example.com>
From: <sip:verona@chat.example.com>;gr=JuliC
DateTime: 2008-10-15T15:02:31-03:00
Content-Type: text/plain

O Romeo, Romeo! wherefore art thou Romeo?
-------a786hjs2$
```

3.7. Exit Room

If Juliet decides to exit the chatroom, her client sends a presence stanza of type "unavailable" to the occupant JID she is currently using in the room (here <verona@chat.example.org/JuliC>).

Example: (F17) Juliet exits room

```
<presence from='juliet@example.com/balcony'
          to='verona@chat.example.org/JuliC'
          type='unavailable'/>
```

Upon receiving such a stanza, the XMPP-to-SIP gateway terminates the SIP session by sending a SIP BYE to the MSRP conference server. The MSRP conference server then responds with a 200 OK.

Juliet MAY include a custom exit message in the presence stanza of type "unavailable", in which case it SHOULD be broadcasted to other participants using the methods described above.

Example: (F17) Juliet exits the chatroom

```
<presence from='juliet@example.com/balcony'
          to='verona@chat.example.org/JuliC'
          type='unavailable'>
  <status>Time to go!</status>
</presence>
```

4. MSRP Multi-party Messaging Session to XMPP MUC

This section describes how to map a Multi-party Instant Message (IM) MSRP session to an XMPP Multi-User Chat (MUC) session.

```
SIP User                  Gateway                    XMPP MUC
   |                         |                          |
   |(F1)(SIP) INVITE         |                          |
   |------------------------>|                          |
   |(F2) (SIP) 200 OK        |                          |
   |<------------------------|                          |
   |(F3) (SIP) ACK           |                          |
   |------------------------>|                          |
   |(F4) (MSRP) NICKNAME     |                          |
   |------------------------>|                          |
   |                         |(F5)(XMPP) Enter room     |
   |                         |------------------------->|
   |(F6) (MSRP) 200 OK       |                          |
   |<------------------------|                          |
   |                         |(F7)(XMPP) (XMPP) Presence|
   |                         |<-------------------------|
   |                         |                          |
   |(F8)(SIP) SUBSCRIBE      |                          |
   |------------------------>|                          |
   |     Event:conference    |                          |
   |                         |                          |
   |(F9) (SIP) 200 OK        |                          |
   |<------------------------|                          |
   |(F10) (SIP) NOTIFY       |                          |
   |<------------------------|                          |
   |(F11) (SIP) 200 OK       |                          |
   |------------------------>|                          |
   |                         |(F12)(XMPP) (XMPP) Subject|
   |                         |<-------------------------|
   |                         |                          |
   |(F13)(MSRP) SEND         |                          |
   |------------------------>|                          |
   |(F14)(MSRP) SEND         |                          |
   |------------------------>|                          |
   |                         |(F15)(XMPP) Chat message  |
   |(F16)(MSRP) 200 OK       |------------------------->|
   |<------------------------|(F17)(XMPP) Chat message  |
   |                         |<-------------------------|
   |(F18)(MSRP) SEND         |                          |
   |<------------------------|                          |
   |(F19)(MSRP) 200 OK       |                          |
   |------------------------>|                          |
   .                         .                          .
```

```
      .                         .                         .
      |                         |                         |
      |(F20)(SIP) BYE           |                         |
      |------------------------>|                         |
      |                         |(F21)(XMPP) Exiting a room|
      |                         |------------------------->|
      |(F22)(SIP) 200 OK        |                         |
      |<------------------------|                         |
```

Note: If the XMPP presence stanza is received before the SIP SUBSCRIBE dialog is established for the "conference" event, then the server SHOULD cache the participants list until the subscription is established and delivered in a SIP NOTIFY request.

4.1. Enter Room

When the SIP user ("Romeo") wants to join a groupchat room ("Verona"), he first has to start the SIP session by sending out a SIP INVITE request containing an offered session description that includes an MSRP media line accompanied by a mandatory "path" and "chatroom" attributes. The MSRP media line is also accompanied by an "accept-types" attribute specifing support for a Message/CPIM top level wrapper for the MSRP message.

Example: (F1) SIP user starts the session

```
INVITE sip:verona@chat.example.org SIP/2.0
To: <sip:verona@chat.example.org>
From: "Romeo" <sip:romeo@example.com>;gr=orchard
Call-ID: 742510no
Content-Type: application/sdp
Content-Length: [length]

c=IN IP4 s2x.example.net
m=message 7313 TCP/MSRP *
a=accept-types:message/cpim text/plain text/html
a=path:msrp://s2x.example.net:7313/ansp71weztas;tcp
a=chatroom
```

Upon receiving the INVITE, the SIP-to-XMPP gateway needs to determine the identity of the remote domain, which it does by performing one or more DNS SRV lookups [RFC2782]. The SIP-to-XMPP gateway SHOULD resolve the address present in the To header of the INVITE to an 'im' URI, then follow the rules in [RFC3861] regarding the "_im" SRV service for the target domain contained in the To header. If SRV address resolution fails for the "_im" service, the SIP-to-XMPP gateway MAY attempt a lookup for the "_xmpp-server" service as specified in [RFC6120] or MAY return an error to the sender (i.e.,

502 Bad Gateway).

If SRV address resolution succeeds, the SIP-to-XMPP gateway SHOULD answer successfuly with a SIP 200 OK (F2).

Implementations MAY wait until the nickname is set with an MSRP NICKNAME chunk before joining the XMPP MUC or MAY choose a temporary nickname (such as the SIP From header display name) and use it to join the room.

```
SIP/2.0 200 OK
To: <sip:verona@chat.example.org>
From: "Romeo" <sip:romeo@example.com>;gr=orchard
Contact: <sip:x2s.example.com;transport=tcp> \
         ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL,SUBSCRIBE,NOTIFY"\
         ;automata;isfocus;message;event="conference"
Call-ID: 742510no
Content-Type: application/sdp

c=IN IP4 x2s.example.com
m=message 8763 TCP/MSRP *
a=accept-types:message/cpim text/plain text/html
a=path:msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
a=chatroom:nickname private-messages
```

Example: (F4) MSRP user sets up nickname

```
MSRP a786hjs2 NICKNAME
To-Path: msrp://s2x.example.net:7313/ansp71weztas;tcp
From-Path: msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
Use-Nickname: "Romeo"
    -------a786hjs2
```

Upon receiving the MSRP NICKNAME request, the SIP-to-XMPP gateway is responsible for generating an XMPP presence stanza and sending it to the chatroom.

Example: (F5) Romeo enters chatroom

```
<presence from='romeo@example.com'
          to='verona@chat.example.org/Romeo'>
  <x xmlns='http='http://jabber.org/protocol/muc'/>
</presence>
```

If the room does not already contain another user with the requested nickname, the service accepts the access request. Thus if the gateway does not receive any stanza of type "error" specifying a <conflict/> error condition, it MUST answer the MSRP nickname

proposal with a 200 OK response (F6).

Example: (F6) Acknowledgement of join

```
MSRP a786hjs2 200 OK
To-Path: msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
From-Path: msrp://s2x.example.net:7313/ansp71weztas;tcp
    -------a786hjs2
```

So far we have assumed that the requested nickname did not conflict with any existing nicknames. The following flow shows the handling of a nickname conflict.

```
                                                XMPP conference
SIP User                  Gateway                   server
   |                        |                          |
   |(F1)(SIP) INVITE        |                          |
   |------------------------>|                          |
   |(F2) (SIP) 200 OK       |                          |
   |<------------------------|                          |
   |(F3) (SIP) ACK          |                          |
   |------------------------>|                          |
   |                        |                          |
   |(F4) (MSRP) NICKNAME    |                          |
   |------------------------>|                          |
   |                        |(F5)(XMPP) Entering a room|
   |                        |-------------------------->|
   |                        |(F7) (XMPP) Presence Error|
   |                        |<--------------------------|
   |(F6) (MSRP) 425 Error   |                          |
   |<------------------------|                          |
   |                        |                          |
```

4.2. Change Nickname

If Romeo decides to change his nickname within the room, he MUST send a new MSRP NICKNAME request. In fact modification of the nickname in MSRP is not different from the initial reservation and usage of a nickname.

```
                                                        XMPP conference
   SIP User                    Gateway                        server
       |                           |                            |
       |(F1) (MSRP) NICKNAME       |                            |
       |-------------------------->|                            |
       |                           |(F2) (XMPP) Presence w/ Nickname
       |                           |--------------------------->|
       |(F3) (MSRP) 200 OK         |                            |
       |<--------------------------|                            |
```

Example: (F1) MSRP user changes nickname

```
MSRP a786hjs2 NICKNAME
To-Path: msrp://s2x.example.net:7313/ansp71weztas;tcp
From-Path: msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
Use-Nickname: "montecchi"
    -------a786hjs2
```

Upon receiving such a message, the SIP-to-XMPP gateway MUST translate it into an XMPP presence stanza.

Example: (F2) XMPP translation of nickname change

```
<presence from='romeo@example.com'
          to='verona@chat.example.org/montecchi'/>
```

4.3. Invite Another User to a Room

To follow.

4.4. Presence Broadcast

If the multi-user chat service is able to add the SIP user to the room, it sends presence from all the existing occupants' room JIDs to the new occupant's full JID, including extended presence information about roles in an <x/> element.

Example: (F7) Chatroom presence information translated into XMPP

```
<presence from='verona@chat.example.org/Romeo'
          to='romeo@example.com'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='none' role='participant'/>
  </x>
</presence>

<presence from='verona@chat.example.org/Ben'
          to='romeo@example.com'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='none' role='participant'/>
  </x>
</presence>
```

Upon receiving these presence stanzas, if the MSRP conference server has already completed the subscription to the Conference Event package [RFC4575], the XMPP-to-SIP gateway MUST translate them in a SIP NOTIFY request containing the participant list (represented in the [RFC4575] format).

Example: (F10) MSRP translation of XMPP participant presence

```
NOTIFY sip:romeo@example.com SIP/2.0
To: <sip:romeo@example.com>;tag=43524545
From: <sip:verona@chat.example.org>;tag=a3343df32
Call-ID: k3l43id034kserrff
Event: conference
Subscription-State: active;expires=3600
Content-Type: application/conference-info+xml
Content-Length: ...

<conference-info version="0" state="full"
    entity="sip:verona@chat.example.org">
  <conference-description>
    <subject>Today in Verona</subject>
    <conf-uris>
      <entry>
        <uri>tel:+18882934234</uri>
        <uri>sip:verona@chat.example.org</uri>
      </entry>
    </conf-uris>
 </conference-description>
 <users>
   <user entity="sip:verona@chat.example.org/JuliC"
         state="full">
     <display-text>JuliC</display-text>
     <roles>
       <entry>participant</entry>
     </roles>
   </user>
   <user entity="sip:verona@chat.example.org/Ben"
         state="full">
     <display-text>Ben</display-text>
     <roles>
       <entry>participant</entry>
     </roles>
 </users>
</conference-info>
```

4.5. Exchange Messages

Once the user has joined the chat room, the user can exchange an unbounded number of messages both public and private.

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

Table 4: Message syntax mapping from MSRP Message to XMPP

CPIM Header	XMPP Element or Attribute
To	to
From	from
body of the SEND request	<body/>

4.5.1. Send a Message to All Occupants

When Romeo wants to send a message to all other occupants in the room, he sends an MSRP SEND request to <room@service> itself (i.e., <verona@chat.example.org> in our example).

The following examples show an exchange of a public message.

Example: (F12) Romeo sends a message to the chat room

```
MSRP a786hjs2 SEND
To-Path: msrp://s2x.example.net:7313/ansp71weztas;tcp
From-Path: msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
Message-ID: 87652492
Byte-Range: 1-*/*
Content-Type: message/cpim

To: <sip:verona@chat.example.org>
From: "Romeo" <sip:romeo@example.com>;gr=orchard
DateTime: 2008-10-15T15:02:31-03:00
Content-Type: text/plain

Romeo is here!
-------a786hjs2$
```

Upon receiving the SEND request, if the request either contains a Failure-Report header field value of "yes" or does not contain a Failure-Report header at all, the SIP-to-XMPP gateway MUST immediately translate it into an XMPP message stanza (F13) and then generate and send an MSRP response (F14).

Example: (F13) XMPP translation of message

```
<message from='romeo@example.com/orchard'
         to='verona@chat.example.org'
         type='groupchat'
         id='8gbx1g4p'>
  <body>Romeo is here!</body>
</message>
```

Example: (F14) MSRP response to public message

```
MSRP d93kswow 200 OK
To-Path: msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
From-Path: msrp://s2x.example.net:7313/ansp71weztas;tcp
-------d93kswow$
```

Note well that the XMPP MUC room will reflect the sender's message back to all users, including the sender. In MSRP this reflected message is unnecessary. Therefore gateways are advised to maintain a cache and if the same stanza is received within a reasonable amount of time, assume is the reflected message and ignore it.

4.5.2. Send a Private Message

Romeo can send a "private message" to a selected occupant via the chat room service by sending a message to the occupant's room nickname.

The following examples show an exchange of a private message.

Example: (F12) Romeo sends a private message

```
MSRP a786hjs2 SEND
To-Path: msrp://s2x.example.net:7313/ansp71weztas;tcp
From-Path: msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
Message-ID: 87652492
Byte-Range: 1-*/*
Content-Type: message/cpim

To: <sip:verona@chat.example.org>;gr=JuliC
From: "Romeo" <sip:romeo@example.com>;gr=orchard
DateTime: 2008-10-15T15:02:31-03:00
Content-Type: text/plain

I am here!!!
-------a786hjs2$
```

The MSRP conference is responsible for transforming the "From"

address into an in-room address.

Example: MSRP handling of private message

```
MSRP a786hjs2 SEND
To-Path: msrp://s2x.example.net:7313/ansp71weztas;tcp
From-Path: msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
Message-ID: 87652492
Byte-Range: 1-*/*
Content-Type: message/cpim

To: <sip:verona@chat.example.org>;gr=JuliC
From: <sip:verona@chat.example.org>;gr=Romeo
DateTime: 2008-10-15T15:02:31-03:00
Content-Type: text/plain

I am here!!!
-------a786hjs2$
```

Once the MSRP conference sends that message to the gateway, the gateway is responsible for translating it into XMPP syntax.

Example: (F13) XMPP translation of private message

```
<message from='verona@chat.example.org/Romeo'
         to='verona@chat.example.org/JuliC'
         type='chat'
         id='rg2ca9k7'/>
      <body>I am here!!!</body>
</message>
```

4.6. Exit Room

If Romeo decides to exit the chat room, his client sends a SIP BYE to the <verona@chat.example.org> chat room.

Example: (F11) Romeo terminates the session

```
BYE sip:verona@chat.example.org SIP/2.0
Max-Forwards: 70
From: "Romeo" <sip:romeo@example.net>;tag=786
To: <sip:verona@chat.example.org>;tag=534
Call-ID: 742510no
Cseq: 1 BYE
Content-Length: 0
```

Upon receiving the SIP BYE, the SIP-to-XMPP gateway translates it in a presence stanza (F19) and sends it to the XMPP MUC room service.

Then the SIP-to-XMPP gateway responds with a 200 OK to the MSRP user.

Example: (F19) Romeo exits the chatroom

```
<presence from='romeo@example.com'
          to='verona@chat.example.org/Romeo'
          type='unavailable'>
</presence>
```

5. Handling of Nicknames and Display Names

Fundamental rules for mapping addresses between XMPP and SIP are provided in [I-D.ietf-stox-core]. However, chatrooms include a more specialized, unique identifier for each participant in a room, called a nickname. Implementations are strongly encouraged to apply the rules for preparation and comparison of nicknames specified in [I-D.ietf-precis-nickname].

In addition to nicknames, some groupchat implementations also include display names (which might or might not be different from users' nicknames). A display name need not be unique within the context of a room but instead simply provides a user-friendly name for a participant.

In SIP, the nickname is the value of the XCON 'nickname' attribute of the <user/> element [RFC6501] and the display name is the XML character data of the conference-info <display-text/> element [RFC4575]. In XMPP, the nickname is the value of the resourcepart of the Occupant JID [XEP-0045] and the display name is the XML character data of the <nick/> element [XEP-0172].

In practice, the <display-text/> element is treated as canonical in SIP implementations, and the <nick/> element is rarely used in XMPP implementations. Therefore, for display purposes SIP implementations ought to use the <display-text/> element (not the XCON 'nickname' attribute) and XMPP implementations ought to use the resourcepart of the Occupant JID (not the character data of the <nick/> element).

If there is a conflict between the SIP nickname and the XMPP nickname, the SIP-to-XMPP or XMPP-to-SIP gateway is responsible for adjusting the nickname to avoid the conflict and for informing the SIP or XMPP client of the unique nickname used to join the chatroom.

6. Security Considerations

The security considerations of [RFC3261], [RFC4975], [RFC6120],

[I-D.ietf-stox-core], [I-D.ietf-simple-chat], and [XEP-0045] apply.

Additional security considerations will be provided in a future version of this specification.

7. IANA Considerations

This document requests no actions of the IANA.

8. References

8.1. Normative References

[I-D.ietf-precis-nickname]
Saint-Andre, P., "Preparation and Comparison of Nicknames", draft-ietf-precis-nickname-05 (work in progress), November 2012.

[I-D.ietf-simple-chat]
Niemi, A., Garcia-Martin, M., and G. Sandbakken, "Multi-party Instant Message (IM) Sessions Using the Message Session Relay Protocol (MSRP)", draft-ietf-simple-chat-18 (work in progress), January 2013.

[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-05 (work in progress), June 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.

[RFC3861] Peterson, J., "Address Resolution for Instant Messaging and Presence", RFC 3861, August 2004.

[RFC4579] Johnston, A. and O. Levin, "Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents", RFC 4579, August 2006.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[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.

[XEP-0045] Saint-Andre, P., "Multi-User Chat", XSF XEP 0045, July 2008.

8.2. Informative References

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC3515] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

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

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.

[RFC6501] Novo, O., Camarillo, G., Morgan, D., and J. Urpalainen, "Conference Information Data Model for Centralized Conferencing (XCON)", RFC 6501, March 2012.

[XEP-0172] Saint-Andre, P. and V. Mercier, "User Nickname", XSF XEP 0172, March 2012.

[XEP-0249] Saint-Andre, P., "Direct MUC Invitations", XSF XEP 0249, September 2011.

Appendix A. Acknowledgements

Special thanks to Fabio Forno for his co-authorship of an early version of this document.

Some text in this document was borrowed from [I-D.ietf-stox-core] and from [XEP-0045].

Authors' Addresses

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3282
Email: psaintan@cisco.com

Saul Ibarra Corretge
AG Projects
Dr. Leijdsstraat 92
Haarlem 2021RK
The Netherlands

Email: saul@ag-projects.com

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Salvatore.Loreto@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2014

P. Saint-Andre
Cisco Systems, Inc.
A. Houri
IBM
J. Hildebrand
Cisco Systems, Inc.
July 1, 2013

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

draft-ietf-stox-im-00

Abstract

This document defines a bidirectional protocol mapping for the exchange of single instant messages 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 3
2. Terminology 4
3. XMPP to SIP 4
4. SIP to XMPP 6
5. Content Types 8
6. Security Considerations 8
7. IANA Considerations 8
8. References 9
8.1. Normative References 9
8.2. Informative References 9
Appendix A. Acknowledgements 10
Authors' Addresses 10

1. Introduction

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

- o Various extensions to the Session Initiation Protocol ([RFC3261]) for instant messaging, as developed within the SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) Working Group; the relevant specification for instant messaging is [RFC3428]
- 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]; that is the approach taken by [I-D.ietf-simple-cpim-mapping] and [RFC3922]. By contrast, 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).

Both XMPP and IM-aware SIP systems enable entities to exchange "instant messages". The term "instant message" usually refers to messages sent between two entities for delivery in close to real time (rather than messages that are stored and forwarded to the intended recipient upon request). This document covers single messages only (sometimes called "pager-mode" messaging), since they form the lowest common denominator for instant messaging. One-to-one chat sessions and multi-party groupchat are covered in separate documents.

The architectural assumptions underlying such direct mappings are provided in [I-D.ietf-stox-core], including mapping of addresses and error condisions. The mappings specified in this document cover basic instant messaging functionality, i.e., the exchange of a single instant message between a SIP user and an XMPP user in either direction. Mapping of more advanced functionality is out of scope for this document, but other documents in this "series" cover such topics.

The discussion venue for this document is the mailing list of the STOX WG; visit https://www.ietf.org/mailman/listinfo/stox for subscription information and discussion archives.

2. Terminology

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].

3. XMPP to SIP

As described in [RFC6121], a single instant message is an XML <message/> stanza of type "normal" sent over an XML stream (since "normal" is the default for the 'type' attribute of the <message/> stanza, the attribute is often omitted). In this document we will assume that such a message 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 (this is a simplifying assumption introduced for explanatory purposes only; the XMPP sender could be a bot-controlled client, a component such as a workflow application, a server, etc.). Continuing the tradition of Shakespeare examples in XMPP documentation, we will say that the XMPP user has an XMPP address of <juliet@example.com>.

When Juliet wants to send an instant message to Romeo, she interacts with her XMPP client, which generates an XMPP <message/> stanza. The syntax of the <message/> stanza, including required and optional elements and attributes, is defined in [RFC6121]. The following is an example of such a stanza:

Example: XMPP user sends message:

```
|  <message from='juliet@example.com/balcony'
|           to='romeo@example.net'>
|    <body>Art thou not Romeo, and a Montague?</body>
|  </message>
```

Upon receiving such a stanza, the XMPP server to which Juliet has connected either delivers it to a local recipient (if the hostname in the 'to' attribute matches one of the hostnames serviced by the XMPP server) or attempts to route it to the foreign domain that services the hostname in the 'to' attribute. Naturally, in this document we assume that the hostname in the 'to' attribute is an IM-aware SIP service hosted by a separate server. As specified in [RFC6121], the XMPP server needs to determine the identity of the foreign domain, which it does by performing one or more DNS SRV lookups [RFC2782]. For message stanzas, the order of lookups recommended by [RFC6121] is to first try the "_xmpp-server" service as specified in [RFC6120] and to then try the "_im" service as specified in [RFC3861]. Here we

assume that the first lookup will fail but that the second lookup will succeed and return a resolution "_im._simple.example.net.", since we have already assumed that the example.net hostname is running a SIP instant messaging service. (Note: The XMPP server may have previously determined that the foreign domain is a SIMPLE server, in which case it would not need to perform the SRV lookups; the caching of such information is a matter of implementation and local service policy, and is therefore out of scope for this document.)

Once the XMPP server has determined that the foreign domain is serviced by a SIMPLE server, it must determine how to proceed. We here assume that the XMPP server contains or has available to it an XMPP-SIMPLE gateway (such an architecture is described in [I-D.ietf-stox-core]). The XMPP server would then deliver the message stanza to the XMPP-SIMPLE gateway.

The XMPP-SIMPLE gateway is then responsible for translating the XMPP message stanza into a SIP MESSAGE request from the XMPP user to the SIP user:

Example: XMPP user sends message (SIP transformation):

```
|  MESSAGE sip:romeo@example.net SIP/2.0
|  Via: SIP/2.0/TCP x2s.example.com;branch=z9hG4bK776sgdkse
|  Max-Forwards: 70
|  To: sip:romeo@example.net
|  From: sip:juliet@example.com
|  Contact: sip:juliet@example.com;gr=balcony
|  Call-ID: Hr0zny9l3@example.com
|  CSeq: 1 MESSAGE
|  Content-Type: text/plain
|  Content-Length: 35
|
|  Art thou not Romeo, and a Montague?
```

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 4: Message syntax mapping from XMPP to SIP

XMPP Element or Attribute	SIP Header or Contents
<body/>	body of MESSAGE
<subject/>	Subject
<thread/>	Call-ID
from	From
id	(no mapping)
to	To
type	(no mapping)
xml:lang	Content-Language

4. SIP to XMPP

As described in [RFC3428], a single instant message is a SIP MESSAGE request sent from a SIP user agent to an intended recipient who is most generally referenced by an Instant Message URI of the form <im:user@domain> but who may be referenced by a SIP or SIPS URI of the form <sip:user@domain> or <sips:user@domain>. Here again we introduce the simplifying assumption that the user agent is controlled by a human user, whom we shall dub <romeo@example.net>.

When Romeo wants to send an instant message to Juliet, he interacts with his SIP user agent, which generates a SIP MESSAGE request. The syntax of the MESSAGE request is defined in [RFC3428]. The following is an example of such a request:

Example: SIP user sends message:

```
|  MESSAGE sip:juliet@example.com SIP/2.0
|  Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKeskdgs677
|  Max-Forwards: 70
|  To: sip:juliet@example.com;gr=balcony
|  From: sip:romeo@example.net
|  Contact: sip:romeo@example.net;gr=orchard
|  Call-ID: M4spr4vdu@example.net
|  CSeq: 1 MESSAGE
|  Content-Type: text/plain
|  Content-Length: 44
|
|  Neither, fair saint, if either thee dislike.
```

Section 5 of [RFC3428] stipulates that a SIP User Agent presented with an im: URI should resolve it to a sip: or sips: URI. Therefore

we assume that the Request-URI of a request received by a SIMPLE-XMPP gateway will contain a sip: or sips: URI. The gateway SHOULD resolve that address to an im: URI for SIP MESSAGE requests, then follow the rules in [RFC3861] regarding the "_im" SRV service for the target domain contained in the Request-URI. If SRV address resolution fails for the "_im" service, the gateway MUST either attempt a lookup for the "_xmpp-server" service as specified in [RFC6120] or return an error to the sender (the SIP "502 Bad Gateway" error seems most appropriate; see [I-D.ietf-stox-core] for details). If SRV address resolution succeeds, the gateway is responsible for translating the request into an XMPP message stanza from the SIP user to the XMPP user and returning a SIP "200 OK" message to the sender:

Example: SIP user sends message (XMPP transformation):

```
|  <message from='romeo@example.net/orchard'
|           to='juliet@example.com/balcony'>
|    <body>Neither, fair saint, if either thee dislike.</body>
|  </message>
```

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

Table 5: Message syntax mapping from SIP to XMPP

SIP Header or Contents	XMPP Element or Attribute
Call-ID	<thread/>
Content-Language	xml:lang
CSeq	(no mapping)
From	from
Subject	<subject/>
Request-URI	to
body of MESSAGE	<body/>

Note: When transforming SIP pager-mode messages, a SIMPLE-XMPP gateway SHOULD specify no XMPP 'type' attribute or, equivalently, a 'type' attribute whose value is "normal".

Note: See Section 5 of this document about the handling of SIP message bodies that contain content types other than plain text.

5. Content Types

SIP requests of type MESSAGE are allowed to contain essentially any content type. The recommended procedures for SIMPLE-to-XMPP gateways to use in handling these content types are as follows.

A SIMPLE-to-XMPP gateway MUST process SIP messages that contain message bodies of type "text/plain" and MUST encapsulate such message bodies as the XML character data of the XMPP <body/> element.

A SIMPLE-to-XMPP gateway SHOULD process SIP messages that contain message bodies of type "text/html"; if so, a gateway MUST transform the "text/html" content into XHTML content that conforms to the XHTML 1.0 Integration Set specified in [XEP-0071].

Although a SIMPLE-to-XMPP gateway MAY process SIP messages that contain message bodies of types other than "text/plain" and "text/html", the handling of such content types is a matter of implementation.

6. Security Considerations

Detailed security considerations for instant messaging protocols are given in [RFC2779], for SIP-based instant messaging in [RFC3428] (see also [RFC3261]), and for XMPP-based instant messaging in [RFC6121] (see also [RFC6120]).

This document specifies methods for exchanging instant messages through a gateway that translates between SIP and XMPP. Such a gateway MUST be compliant with the minimum security requirements of the instant messaging protocols for which it translates (i.e., SIP and XMPP). The addition of gateways to the security model of instant messaging specified in [RFC2779] introduces some new risks. In particular, end-to-end security properties (especially confidentiality and integrity) between instant messaging user agents that interface through a SIMPLE-XMPP gateway can be provided only if common formats are supported. Specification of those common formats is out of scope for this document, although it is preferred to use [RFC3862] for instant messages.

7. IANA Considerations

This document requests no actions of IANA.

8. References

8.1. 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-00 (work in progress), July 2013.

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

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[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.

[RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[RFC3861] Peterson, J., "Address Resolution for Instant Messaging and Presence", RFC 3861, 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.

[XEP-0071]
Saint-Andre, P., "XHTML-IM", XSF XEP 0071, November 2012.

8.2. 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.

[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.

[RFC3862] Klyne, G. and D. Atkins, "Common Presence and Instant Messaging (CPIM): Message Format", RFC 3862, 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.

Appendix A. Acknowledgements

The authors wish to thank the following individuals for their feedback: Adrian Georgescu, Saul Ibarra, Salvatore Loreto, and Tory Patnoe.

Authors' Addresses

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3282
Email: psaintan@cisco.com

Avshalom Houri
IBM
Building 18/D, Kiryat Weizmann Science Park
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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2014

P. Saint-Andre
Cisco Systems, Inc.
A. Houri
IBM
J. Hildebrand
Cisco Systems, Inc.
July 1, 2013

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

draft-ietf-stox-presence-00

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 3
2. Terminology 3
3. Presence Subscriptions 3
4. Presence Notifications 11
5. Security Considerations 17
6. IANA Considerations 18
7. References 18
7.1. Normative References 18
7.2. Informative References 19
Appendix A. Acknowledgements 19
Authors' Addresses 19

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 instant messaging, as developed within the SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) Working Group; the relevant specification for presence is [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]; that is the approach taken by both [RFC3922] and [I-D.ietf-simple-cpim-mapping]. 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).

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.

The discussion venue for this document is the mailing list of the STOX WG; visit https://www.ietf.org/mailman/listinfo/stox for subscription information and discussion archives.

2. Terminology

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].

3. Presence Subscriptions

3.1. 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 "none" (neither the user nor the contact is subscribed to the other's presence information), "from" (the user has a subscription from the contact), "to" (the user has a subscription to the contact's presence information), and "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 an Instant Message 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>.

The subscription models underlying XMPP and SIP are quite different. For instance, XMPP presence subscriptions are long-lived (indeed permanent if not explicitly cancelled), 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]). These differences are addressed below.

3.2. XMPP to SIP

3.2.1. Establishing

An XMPP user (e.g., juliet@example.com) initiates a subscription by sending a subscription request to another entity (e.g., romeo@example.net), and the other entity (conventionally called a "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: XMPP user subscribes to SIP contact:

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

Upon receiving such a stanza, the XMPP server to which the user has connected needs to determine the identity of the foreign domain, which it does by performing one or more DNS SRV lookups [RFC2782]. For presence stanzas, the order of lookups recommended by [RFC6121] is to first try the "_xmpp-server" service as specified in [RFC6120] and to then try the "_pres" service as specified in [RFC3861]. Here we assume that the first lookup will fail but that the second lookup will succeed and return a resolution "_pres._simple.example.net.", since we have already assumed that the example.net hostname is running a SIP presence service.

Once the XMPP server has determined that the foreign domain is serviced by a SIMPLE server, it needs to determine how to proceed. We here assume that the XMPP server contains or has available to it an XMPP-SIMPLE gateway or connection manager (which enables it to speak natively to SIMPLE servers). The XMPP server would then deliver the presence stanza to the XMPP-SIMPLE gateway.

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

Example: XMPP user subscribes to SIP contact (SIP transformation):

```
|  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: l04th3s1p@example.com
|  Event: presence
|  Max-Forwards: 70
|  CSeq: 123 SUBSCRIBE
|  Contact: <sip:sipgate.example.com;transport=tcp>
|  Accept: application/pidf+xml
|  Expires: 3600
|  Content-Length: 0
```

The SIP user then SHOULD send a response indicating acceptance of the subscription request:

Example: SIP accepts subscription request:

```
|  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: l04th3s1p@example.com
|  CSeq: 234 SUBSCRIBE
|  Contact: <sip:simple.example.net;transport=tcp>
|  Expires: 3600
|  Content-Length: 0
```

In accordance with [RFC6665], the XMPP-SIMPLE 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 4).

Example: 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
|  Call-ID: l04th3s1p@example.com
|  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>
```

Upon receiving the first NOTIFY with a subscription state of active, the XMPP-SIMPLE gateway MUST generate a presence stanza of type "subscribed":

Example: XMPP user receives acknowledgement from SIP contact:

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

As described under Section 4, the gateway MUST also generate a presence notification to the XMPP user:

Example: XMPP user receives presence notification from SIP contact:

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

3.2.2. Refreshing

It is the responsibility of the XMPP-SIMPLE 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 (e.g., the XMPP-SIMPLE gateway SHOULD send a new SUBSCRIBE request to the SIP user whenever the XMPP user sends initial presence to its XMPP server, i.e., upon initiating a presence session with the XMPP server). See the Security Considerations (Section 5) of this document for important information and requirements regarding the security implications of this functionality.

3.2.3. Cancelling

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: XMPP user unsubscribes from SIP contact:

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

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

Example: XMPP user unsubscribes from SIP contact (SIP transformation):

```
|  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: 1ckm32@example.com
|  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-SIMPLE gateway SHOULD a presence stanza of type "unsubscribed" to the XMPP user:

Example: XMPP user receives unsubscribed notification:

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

3.3. SIP to XMPP

3.3.1. Establishing

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: SIP user subscribes to XMPP contact:

```
|  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: 4wcm0n@example.net
|  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 such a request, a SIMPLE-XMPP gateway is responsible for translating it into an XMPP subscription request from the SIP user to the XMPP user:

Example: SIP user subscribes to XMPP contact (XMPP transformation):

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

In accordance with [RFC6121], 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, discard the subscribe request). If the XMPP user approves the subscription request, the XMPP server then MUST return a presence stanza of type "subscribed" from the XMPP user to the SIP user; if a subscription already exists, the XMPP server SHOULD auto-reply with a presence stanza of type "subscribed". In any case, if the SIMPLE-XMPP gateway receives a presence stanza of type "subscribed" from the XMPP user, it SHOULD silently discard the stanza.

3.3.2. Refreshing

It is the responsibility of the SIMPLE-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 rather odd 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", including a Subscription-State of "terminated" 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 the SIMPLE-XMPP gateway chooses is up to the implementation.

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

Example: SIP subscription expires (treated as temporary by gateway):

```
|  <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: SIP subscription expires (treated as long-lived by gateway):

```
|  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: j4s0h4vny@example.com
|  Event: presence
|  Subscription-State: terminated;reason=timeout
|  Max-Forwards: 70
|  CSeq: 232 NOTIFY
|  Contact: <sip:sipgate.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: SIP subscription expires (treated as long-lived by gateway):

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

3.3.3. Cancelling

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: SIP user cancels subscription:

```
|  SUBSCRIBE 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
|  Call-ID: 1tsn1ce@example.net
|  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 SIMPLE-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.

4. Presence Notifications

4.1. Overview

Both XMPP and presence-aware SIP systems enable entities (often but not necessarily human users) to send presence notifications to other entities. At a minimum, the term "presence" refers to information about an entity’s availability for communication on a network (on/off), often supplemented by information that further specifies the entity’s communications context (e.g., "do not disturb"). Some systems and protocols extend this notion 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 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], 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 (again, this is a simplifying assumption introduced for explanatory purposes only). In general, XMPP presence is sent by the user to the user’s server and then broadcasted 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 Instant Message 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>. Here again we introduce the simplifying assumption that the user agent is controlled by a human user.

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].

4.2. 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: 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. Naturally, in this document we assume that the hostname is a SIP

presence service hosted by a separate server. As specified in [RFC6121], the XMPP server needs to determine the identity of the foreign domain, which it does by performing one or more DNS SRV lookups [RFC2782]. For presence stanzas, the order of lookups recommended by [RFC6121] is to first try the "_xmpp-server" service as specified in [RFC6120] and to then try the "_pres" service as specified in [RFC3861]. Here we assume that the first lookup will fail but that the second lookup will succeed and return a resolution "_pres._simple.example.net.", since we have already assumed that the example.net hostname is running a SIP presence service. (Note: The XMPP server might have previously determined that the foreign domain is a SIMPLE server, e.g., when it sent a SIP SUBSCRIBE to the SIP user when Juliet sent initial presence to the XMPP server, in which case it would not need to perform the SRV lookups; the caching of such information is a matter of implementation and local service policy, and is therefore out of scope for this document.)

Once the XMPP server has determined that the foreign domain is serviced by a SIMPLE server, it needs to determine how to proceed. We here assume that the XMPP server contains or has available to it an XMPP-SIMPLE gateway. The XMPP server would then deliver the presence stanza to the XMPP-SIMPLE gateway.

The XMPP-SIMPLE 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: XMPP user sends presence notification (SIP transformation):

```
|  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: j4s0h4vny@example.com
|  Event: presence
|  Subscription-State: active;expires=599
|  Max-Forwards: 70
|  CSeq: 157 NOTIFY
|  Contact: <sip:sipgate.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 6: 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	Call-ID
to	To
type	basic status [3] [4]
xml:lang	Content-Language
<priority/>	PIDF priority for tuple
<show/>	no mapping [5]
<status/>	<note/>

Note the following regarding these mappings:

1. Only a presence stanza that lacks a ’type’ attribute or whose ’type’ attribute has a value of "unavailable" SHOULD be mapped by an XMPP-SIMPLE 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. 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 for communications, the gateway SHOULD map that information to a PIDF <basic/> status of "closed".
4. When the XMPP-SIMPLE 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).
5. 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.

4.3. 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: 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: j0sj4sv1m@example.net
|  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 such a request, a SIMPLE-XMPP gateway is responsible for translating it into an XMPP presence stanza from the SIP user to the XMPP user:

Example: SIP user sends presence notification (XMPP transformation):

```
|  <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 7: 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 (OPTIONAL)
From	from
priority for tuple	<priority/>
To	to
<note/>	<status/>

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".

5. 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. To help prevent such an attack, access to an XMPP-SIMPLE gateway that is hosted on the XMPP network SHOULD be restricted to XMPP users associated 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-SIMPLE gateway from users who are not associated with a domain that is so related to the hostname of the gateway, it MAY (based on local service provisioning) refuse to service such users or refuse to communicate with the gateway. Furthermore, whenever an XMPP-SIMPLE gateway seeks to refresh an XMPP user’s long-lived subscription to a SIP user’s presence, it MUST first send an XMPP <presence/> 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]; however, the administrator of an XMPP-SIMPLE gateway MAY (based on local service provisioning) exempt "known good" XMPP servers from this check (e.g., the XMPP

server associated with the XMPP-SIMPLE gateway as described above).

6. IANA Considerations

This document requests no actions of IANA.

7. References

7.1. 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-00 (work in progress), July 2013.

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

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[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.

[RFC3861] Peterson, J., "Address Resolution for Instant Messaging and Presence", RFC 3861, 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.

7.2. 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.

[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, Saul Ibarra, Salvatore Loreto, Michael Lundberg, Daniel-Constantin Mierla, and Tory Patnoe.

Authors' Addresses

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3282
Email: psaintan@cisco.com

Avshalom Houri
IBM
Building 18/D, Kiryat Weizmann Science Park
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