

DISPATCH
Internet-Draft
Intended status: Standards Track
Expires: April 17, 2014

R. Ejzak
J. Marcon
Alcatel-Lucent
October 14, 2013

SDP-based WebRTC data channel negotiation
draft-ejzak-dispatch-webrtc-data-channel-sdpneg-00

Abstract

The Real-Time Communication in WEB-browsers (RTCWeb) working group is charged to provide protocols to support direct interactive rich communication using audio, video, and data between two peers' web-browsers. For the support of data communication, the RTCWeb working group has in particular defined the concept of bi-directional data channels over SCTP, where each data channel might be used to transport other protocols, called sub-protocols. Data channel setup can be done using either the in-band WebRTC Data Channel protocol or some external (in-band or out-of-band) negotiation. This document specifies how the SDP offer/answer exchange can be used to achieve such an external negotiation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 17, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions	3
3. Terminology	3
4. WebRTC Data Channels	4
4.1. Stream identifier numbering	4
4.2. Generic external negotiation	5
4.2.1. Overview	5
4.2.2. Opening a data channel	5
4.2.3. Closing a data channel	6
5. SDP-based external negotiation	6
5.1. SDP syntax	6
5.1.1. SDP attribute for data channel parameter negotiation	7
5.1.1.1. stream parameter	7
5.1.1.2. label parameter	7
5.1.1.3. subprotocol parameter	8
5.1.1.4. max_retr parameter	8
5.1.1.5. max_time parameter	8
5.1.1.6. unordered parameter	8
5.1.2. Sub-protocol specific attributes	9
5.2. Procedures	10
5.2.1. Managing stream identifiers	10
5.2.2. Opening a data channel	10
5.2.3. Closing a data channel	12
6. Security Considerations	12
7. IANA Considerations	12
8. Acknowledgments	12
9. References	13
9.1. Normative References	13
9.2. Informative References	13
Authors' Addresses	14

1. Introduction

The RTCWeb working group has defined the concept of bi-directional data channels running on top of SCTP/DTLS. Each data channel consists of paired SCTP streams sharing the same SCTP Stream Identifier. Data channels are created by endpoint applications through the WebRTC API, and can be used to transport proprietary or well-defined protocols, which in the latter case can be signaled by

the data channel "sub-protocol" parameter, conceptually similar to the WebSocket "sub-protocol". However, apart from the "sub-protocol" value transmitted to the peer, RTCWeb leaves open how endpoint applications can agree on how to instantiate a given sub-protocol on a data channel, and whether it is signaled in-band or out-of-band (or both). In particular, the SDP offer generated by the browser includes no channel-specific information.

This document defines SDP-based out-of-band negotiation procedures to establish WebRTC data channels for transport of well-defined sub-protocols.

2. Conventions

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

3. Terminology

This document uses the following terms:

Data channel: A bidirectional channel consisting of paired SCTP outbound and inbound streams.

Data channel properties: fixed properties assigned to a data channel at the time of its creation. Some of these properties determine the way the browser transmits data on this channel (e.g., stream identifier, reliability, order of delivery...)

External negotiation: data channel negotiation based on out-of-band or in-band mechanisms other than the WebRTC data channel protocol.

In-band: transmission through the peer-to-peer SCTP association.

In-band negotiation: data channel negotiation based on the in-band WebRTC data channel protocol defined in [I-D.ietf-rtcweb-data-protocol].

Out-of-band: transmission through the WebRTC signaling path.

Peer: From the perspective of one of the agents in a session, its peer is the other agent. Specifically, from the perspective of the SDP offerer, the peer is the SDP answerer. From the perspective of the SDP answerer, the peer is the SDP offerer.

Stream identifier: the identifier of the outbound and inbound SCTP streams composing a data channel.

4. WebRTC Data Channels

This section summarizes how WebRTC data channels work in general.

A WebRTC application creates a data channel via the WebRTC Data Channel API, by providing a number of setup parameters (sub-protocol, label, reliability, order of delivery, priority). The application also specifies if the browser takes charge of the in-band negotiation using the WebRTC data protocol, or if the application intends to perform an "external negotiation" using some other in-band or out-of-band mechanism.

In any case, the SDP offer generated by the browser is per [I-D.ietf-mmusic-sctp-sdp]. In brief, it contains one m-line for the SCTP association on top of which data channels will run, and one attribute per protocol assigned to the SCTP ports:

```
m=application 54111 DTLS/SCTP 5000 5001 5002
c=IN IP4 79.97.215.79
a=sctpmap:5000 webrtc-datachannel 16
a=sctpmap:5001 bfcf 2
a=sctpmap:5002 t38 1
```

Note: A WebRTC browser will only create an sctpmap attribute for the webrtc-datachannel protocol, and will not create sctpmap attributes for other protocols such as bfcf or t38. This example shows the hypothetical power of the syntax to support multiplexing of SCTP associations for different protocols on the same DTLS connection.

Note: This SDP syntax does not contain any channel-specific information.

4.1. Stream identifier numbering

Independently from the requested type of negotiation, the application creating a data channel can either pass to the browser the stream identifier to assign to the data channel or else let the browser pick one identifier from the ones unused.

To avoid glare situations, each endpoint can moreover own an exclusive set of stream identifiers, in which case an endpoint can only create a data channel with a stream identifier it owns.

Which set of stream identifiers is owned by which endpoint is determined by convention or other means.

For data channels negotiated in-band, one endpoint owns by convention the even stream identifiers, whereas the other owns the odd stream identifiers, as defined in [I-D.ietf-rtcweb-data-protocol].

For data channels externally negotiated, no convention is defined by default.

4.2. Generic external negotiation

4.2.1. Overview

In-band negotiation only provides for negotiation of data channel transport parameters and does not provide for negotiation of sub-protocol specific parameters. External negotiation can be defined to allow negotiation of parameters beyond those handled by in-band negotiation, e.g., parameters specific to the sub-protocol instantiated on a particular data channel. See Section 5.1.2 for an example of such a parameter.

The following procedures are common to all methods of external negotiation, whether in-band (communicated using proprietary means on an already established data channel) or out-of-band (using SDP or some other protocol associated with the signaling channel).

4.2.2. Opening a data channel

In the case of external negotiation, the endpoint application has the option to fully control the stream identifier assignments. However these assignments have to coexist with the assignments controlled by the browser for the in-band negotiated data channels (if any). It is the responsibility of the application to ensure consistent assignment of stream identifiers.

When the application requests the creation of a new data channel to be set up via external negotiation, the browser creates the data channel locally without sending any DATA CHANNEL OPEN message in-band, and sets the data channel state to Connecting if the SCTP association is not yet established, or sets the data channel state to Open if the SCTP association is already established.

The application then externally negotiates the data channel properties and sub-protocol properties with the peer's application.

[ASSUMPTION] The peer must then symmetrically create a data channel with these negotiated data channel properties. This is the only way for the peer's browser to know which properties to apply when transmitting data on this channel. The browser must allow data channel creation with any non-conflicting stream identifier so that both peers can create the data channel with the same stream identifier.

In case the external negotiation is correlated with an SDP offer/answer exchange that establishes the SCTP association, the SCTP initialization completion triggers each endpoint's browser to change the data channel state from Connecting to Open.

Each application must ensure that a data channel is in the Open state both locally and at the peer prior to sending data. This document includes procedures for doing so that are specific to using SDP offer/answer for external negotiation.

[ACTION ITEM] Determine if these procedures are fully consistent with the data channel design and whether additional clarification is needed in data channel documents to ensure proper support of external negotiation.

4.2.3. Closing a data channel

When the application requests the closing of an externally negotiated data channel, the browser always performs an in-band SSN reset for this channel.

Depending upon the method used for external negotiation and the sub-protocol associated with the data channel, the closing might in addition be signaled to the peer via external negotiation.

5. SDP-based external negotiation

This section defines a method of external negotiation by which two WebRTC endpoints can negotiate data channel-specific and sub-protocol-specific parameters, using the out-of-band SDP offer/answer exchange.

5.1. SDP syntax

Two new SDP attributes are defined to support external negotiation of data channels. The first attribute provides for negotiation of channel-specific parameters. The second attribute provides for negotiation of sub-protocol-specific parameters.

5.1.1.1. SDP attribute for data channel parameter negotiation

Associated with the m line that defines the SCTP association for data channels (defined in Section 4), each SDP offer and answer includes an attribute line that defines the data channel parameters for each data channel to be negotiated. Each attribute line specifies the following parameters for a data channel: Stream Identifier, sub-protocol, label, reliability, order of delivery, and priority. The following is an example of the attribute line for sub-protocol "BFCP" and stream id "2" :

```
a=webrtc-DataChannel:5000 stream=2;label="channel 2"; \
  subprotocol="BFCP";max_retr=3
```

This line MUST be replicated without changes in the SDP answer, if the answerer accepts the offered data channel.

This line MUST be replicated without changes in any subsequent offer or answer, as long as the data channel is still opened at the time of offer or answer generation.

Note: This attribute was defined in old version 03 of the following draft but was removed along with any support for SDP external negotiation in subsequent versions:
[I-D.ietf-mmusic-sctp-sdp].

Note: This document does not provide a complete specification of how to negotiate the use of a data channel to transport BFCP. Procedures specific to each sub-protocol such as BFCP will be documented elsewhere. The use of BFCP is only an example of how the generic procedures described herein might apply to a specific sub-protocol.

5.1.1.1.1. stream parameter

The 'stream' parameter indicates the actual stream identifier within the association used to form the channel. Stream is a mandatory parameter.

```
stream-attr      = "a=stream=" streamidentifier
streamidentifier = 1*DIGIT
```

5.1.1.1.2. label parameter

The 'label' parameter indicates the name of the channel. It represents a label that can be used to distinguish, in the context of

the WebRTC API, an `RTCDataChannel` object from other `RTCDataChannel` objects. Label is a mandatory parameter.

```
label-attr      = "a=label=" labelstring
labelstring     = text
text            = byte-string
```

5.1.1.3. subprotocol parameter

The 'subprotocol' parameter indicates which protocol the client expects to exchange via the channel. Subprotocol is a mandatory parameter.

```
subprotocol-attr = "a=subprotocol=" labelstring
labelstring     = text
text            = byte-string
```

[ACTION ITEM] The IANA registry to be used for the subprotocol parameter is still to be determined. It also needs to be determined what the relationship is to existing registries and how to reference already-existing protocols.

5.1.1.4. max_retr parameter

The 'max_retr' parameter indicates the max times a user message will be retransmitted. The max_retr parameter is optional with default value unbounded.

```
maxretr-attr    = "a=maxretr=" maxretrvalue
maxretrvalue    = 1*DIGIT
```

5.1.1.5. max_time parameter

A user messages will no longer be transmitted or retransmitted after a specified life-time given in milliseconds in the 'max_time' parameter. The max_time parameter is optional with default value unbounded.

```
maxtime-attr    = "a=maxtime=" maxtimevalue
maxtimevalue    = 1*DIGIT
```

5.1.1.6. unordered parameter

The 'unordered' parameter indicates that DATA chunks in the channel MUST be dispatched to the upper layer by the receiver without any attempt to reorder. The unordered parameter is optional. If the unordered parameter is absent, the receiver is required to deliver DATA chunks to the upper layer in proper order.

5.1.2. Sub-protocol specific attributes

In the SDP, each data channel declaration MAY also be followed by other SDP attributes specific to the sub-protocol in use. Each of these attributes is represented by one new attribute line, and it includes the contents of a media-level SDP attribute already defined for use with this (sub)protocol in another IETF specification. Sub-protocol-specific attributes might also be defined for exclusive use with data channel transport, but should use the same syntax described here for other sub-protocol-specific attributes.

Each sub-protocol specific SDP attribute that would normally be used to negotiate the subprotocol using SDP is replaced with an attribute of the form "a=wdcsa:sctp-port:stream-id original-attribute", where wdcsa stands for "webrtc-DataChannel sub-protocol attribute", sctp-port is the sctp port number assigned for webrtc-DataChannel on the media line, stream-id is the sctp stream identifier assigned to this sub-protocol instance, and original-attribute represents the contents of the sub-protocol related attribute to be included.

```
a=webrtc-DataChannel:5000 stream=2;label="channel 2"; \  
  subprotocol="MSRP";max_retr=3  
a=wdcsa:5000:2 accept-types:text/plain
```

Thus in the example above, the original attribute line "a=accept-types:text/plain" is represented by the attribute line "a=wdcsa:5000:2 accept-types:text/plain", which specifies that this instance of MSRP being transported on the sctp association using port number 5000 and the data channel with stream id 2 accepts plain text files.

As opposed to the data channel setup parameters, these parameters are subject to offer/answer negotiation following the procedures defined in the sub-protocol specific documents.

The same syntax applies to any other SDP attribute required for negotiation of this instance of the sub-protocol.

Note: This document does not provide a complete specification of how to negotiate the use of a data channel to transport MSRP. Procedures specific to each sub-protocol such as MSRP will be documented

elsewhere. The use of MSRP is only an example of how the generic procedures described herein might apply to a specific sub-protocol.

5.2. Procedures

5.2.1. Managing stream identifiers

For the SDP-based external negotiation described in this document, the initial offerer (in the context of a `PeerConnection`) owns by convention the even stream identifiers whereas the initial answerer owns the odd stream identifiers. This ownership is invariant for the whole lifetime of the `PeerConnection`, e.g. it does not change if the initial answerer sends a new offer to the initial offerer.

[ACTION ITEM] This convention is different from the convention currently defined for in-band negotiation, where even/odd assignment is determined by DTLS role. Since DTLS role cannot be determined until after the initial SDP offer/answer is complete, this convention cannot be used for external negotiation. It might be appropriate to change the convention for stream identifier assignment for in-band negotiation for consistency with external negotiation. Otherwise it might be necessary to prohibit simultaneous use of in-band and external negotiation for data channels.

5.2.2. Opening a data channel

The procedure for opening a data channel using external negotiation starts with the agent preparing to send an SDP offer. If a peer receives an SDP offer before getting to send a new SDP offer with data channels that are to be externally negotiated, or loses an SDP offer glare resolution procedure in this case, it must wait until the ongoing SDP offer/answer completes before resuming the external negotiation procedure.

The agent that intends to send an SDP offer to create data channels through SDP-based external negotiation performs the following:

- o Creates data channels using stream identifiers from the owned set (see Section 5.2.1).
- o As described in Section 4.2.2, if the SCTP association is not yet established, then the newly created data channels are in the Connecting state, else if the SCTP association is already established, then the newly created data channels are in the Open state.
- o Obtains a new SDP offer from the browser.

- o Determines the list of stream identifiers assigned to data channels opened through external negotiation.
- o Completes the SDP offer with the webrtc-DataChannel and wdcsa attributes needed for each externally-negotiated data channel, as described in Section 5.1.
- o Sends the SDP offer.

The peer receiving such an SDP offer performs the following:

- o Applies the SDP offer. Note that the browser ignores data channel specific attributes in the SDP.
- o Analyzes the channel parameters and sub-protocol attributes to determine whether to accept each offered data channel.
- o For accepted data channels, creates peer instances for the data channels with the browser using the channel parameters described in the SDP offer. Note that the browser is asked to create data channels with stream identifiers not "owned" by the agent.
- o As described in Section 4.2.2, if the SCTP association is not yet established, then the newly created data channels are in the Connecting state, else if the SCTP association is already established, then the newly created data channels are in the Open state.
- o Obtains a new SDP answer from the browser.
- o Completes the SDP answer with the webrtc-DataChannel and wdcsa attributes needed for each externally-negotiated data channel, as described in Section 5.1.
- o Sends the SDP answer.

The agent receiving such an SDP answer performs the following:

- o Closes any created data channels (whether in Connecting or Open state) for which the expected webrtc-DataChannel and wdcsa attributes are not present in the SDP answer.
- o Applies the SDP answer.

Any data channels in Connecting state are transitioned to the Open state when the SCTP association is established.

Each agent application must wait to send data until it has confirmation that the data channel at the peer is in the Open state. This occurs:

- o At both peers when a data channel is created without an established SCTP association, as soon as the browsers report that the data channel transitions to the Open state from the Connecting state.
- o At the agent receiving an SDP offer for which there is an established SCTP association, as soon as it creates an externally negotiated data channel in the Open state based on information signaled in the SDP offer.
- o At the agent sending an SDP offer to create a new externally negotiated data channel for which there is an established SCTP association, when it receives the SDP answer confirming acceptance of the data channel or when it begins to receive data on the data channel from the peer, whichever occurs first.

5.2.3. Closing a data channel

When the application requests the closing of a data channel that was externally negotiated, the browser always performs an in-band SSN reset for this channel.

It is specific to the sub-protocol whether this closing must in addition be signaled to the peer via a new SDP offer/answer exchange.

The application must also close any data channel that was externally negotiated, for which the stream identifiers are not listed in an incoming SDP offer.

6. Security Considerations

To be completed.

7. IANA Considerations

To be completed.

8. Acknowledgments

The authors wish to acknowledge the borrowing of ideas from other internet drafts by Salvatore Loreto, Gonzalo Camarillo, Peter Dunkley and Gavin Llewellyn, and to thank Paul Kyzivat, Jonathan Lennox, Uwe Rauschenbach and Keith Drage for their invaluable comments.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [I-D.ietf-rtcweb-jsep] Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-04 (work in progress), September 2013.
- [I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "RTCWeb Data Channels", draft-ietf-rtcweb-data-channel-05 (work in progress), July 2013.
- [I-D.ietf-mmusic-sctp-sdp] Loreto, S. and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-04 (work in progress), June 2013.
- [WebRtcAPI] Bergkvist, A., Burnett, D., Narayanan, A., and C. Jennings, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD WD-webrtc-20120821, August 2012, <<http://www.w3.org/TR/2012/WD-webrtc-20120821>>.

9.2. Informative References

- [I-D.ietf-rtcweb-data-protocol] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Protocol", draft-ietf-rtcweb-data-protocol-00 (work in progress), July 2013.
- [RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

- [RFC4976] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions for the Message Sessions Relay Protocol (MSRP)", RFC 4976, September 2007.
- [RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009.
- [RFC6135] Holmberg, C. and S. Blau, "An Alternative Connection Model for the Message Session Relay Protocol (MSRP)", RFC 6135, February 2011.
- [RFC6714] Holmberg, C., Blau, S., and E. Burger, "Connection Establishment for Media Anchoring (CEMA) for the Message Session Relay Protocol (MSRP)", RFC 6714, August 2012.

Authors' Addresses

Richard Ejzak
Alcatel-Lucent
1960 Lucent Lane
Naperville, Illinois 60563-1594
US

Phone: +1 630 979 7036
Email: richard.ejzak@alcatel-lucent.com

Jerome Marcon
Alcatel-Lucent
Route de Villejust
Nozay 91620
France

Email: jerome.marcon@alcatel-lucent.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 05, 2014

E. Ivov
Jitsi
P. Saint-Andre
Cisco Systems, Inc.
E. Marocco
Telecom Italia
October 02, 2013

CUSAX: Combined Use of the Session Initiation Protocol (SIP) and the
Extensible Messaging and Presence Protocol (XMPP)
draft-ivov-xmpp-cusax-09

Abstract

This document suggests some strategies for the combined use of the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP) both in user-oriented clients and in deployed servers. Such strategies, which mainly consist of configuration changes and minimal software modifications to existing clients and servers, aim to provide a single, full-featured, real-time communication service by using complementary subsets of features from SIP and from XMPP. Typically such subsets consist of telephony capabilities from SIP and instant messaging and presence capabilities from XMPP. This document does not define any new protocols or syntax for either SIP or XMPP, and by intent does not attempt to standardize "best current practices". Instead, it merely aims to provide practical guidance to those who are interested in the combined use of SIP and XMPP for real-time communication.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 05, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Client Bootstrap	5
3. Operation	6
3.1. Server-Side Setup	7
3.2. Service Management	7
3.3. Client-Side Discovery and Usability	8
3.4. Indicating a Relationship Between SIP and XMPP Accounts	9
3.5. Matching Incoming SIP Calls to XMPP JIDs	10
4. Multi-Party Interactions	11
5. Federation	12
6. Summary of Suggested Strategies	13
7. IANA Considerations	14
8. Security Considerations	14
9. References	15
9.1. Normative References	15
9.2. Informative References	16
Appendix A. Acknowledgements	17
Authors' Addresses	18

1. Introduction

Historically SIP [RFC3261] and XMPP [RFC6120] have often been implemented and deployed with different purposes: from its very start SIP's primary goal has been to provide a means of conducting "Internet telephone calls". XMPP on the other hand, has, from its Jabber days, been mostly used for instant messaging and presence [RFC6121], as well as related services such as groupchat rooms [XEP-0045].

For various reasons, these trends have continued through the years even after each of the protocols had been equipped to provide the features it was initially lacking:

- o In the context of the SIMPLE working group, the IETF has defined a number of protocols and protocol extensions that not only allow for SIP to be used for regular instant messaging and presence but that also provide mechanisms for related features such as multi-party chat, server-stored contact lists, and file transfer [RFC6914].
- o Similarly, the XMPP community and the XMPP Standards Foundation have worked on defining a number of XMPP Extension Protocols (XEPs) that provide XMPP implementations with the means of establishing end-to-end sessions. These extensions are often jointly referred to as Jingle [XEP-0166] and arguably their most popular use case is audio and video calling [XEP-0167].

However, although SIP has been extended for messaging and presence and XMPP has been extended for voice and video, the reality is that SIP remains the protocol of choice for telephony-like services and XMPP remains the protocol of choice for IM and presence services. As a result, a number of adopters have found themselves needing features that are not offered by any single-protocol solution, but that separately exist in SIP and XMPP implementations. The idea of seamlessly using both protocols together would hence often appeal to service providers and users. Most often, such a service would employ SIP exclusively for audio, video, and telephony services and rely on XMPP for anything else varying from chat, contact list management, and presence to whiteboarding and exchanging files. Because these services and clients involve the combined use of SIP and XMPP, we label them "CUSAX" for short.

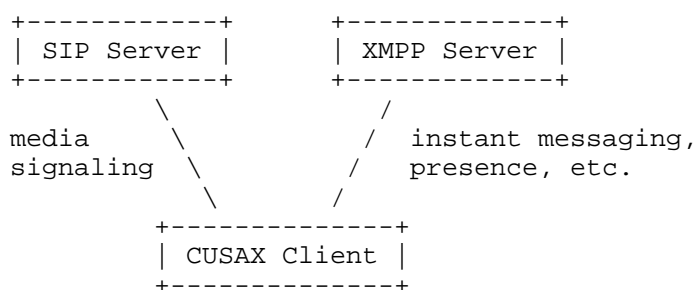


Figure 1: Division of Responsibilities

This document suggests different configuration options and minimal modifications to existing software so that clients and servers can

offer these hybrid services while providing an optimal user experience. It covers server discovery, determining a SIP Address of Record (AOR) while using XMPP, and determining an XMPP Jabber Identifier ("JID") from incoming SIP requests. Most of the text here pertains to client behavior but we also suggest certain server-side configurations and operational strategies. The document also discusses significant security considerations that can arise when offering a dual-protocol solution, and provides advice for avoiding security mismatches that would result in degraded communications security for end users.

Note that this document is focused on coexistence of SIP and XMPP functionality in end-user-oriented clients. By intent it does not define methods for protocol-level mapping between SIP and XMPP, as might be used within a server-side gateway between a SIP network and an XMPP network (a separate series of documents has been produced that defines such mappings). More generally, this document does not describe service policies for inter-domain communication (often called "federation") between service providers (e.g., how a service provider that offers a CUSAX service might communicate with a SIP-only or XMPP-only service), nor does it describe the reasons why a service provider might choose SIP or XMPP for various features.

This document concentrates on use cases where the SIP services and XMPP services are controlled by one and the same provider, since that assumption greatly simplifies both client implementation and server-side deployment (e.g., a single service provider can enforce common or coordinated policies across both the SIP and XMPP aspects of a CUSAX service, which is not possible if a SIP service is offered by one provider and an XMPP service is offered by another provider). Since this document is of an informational nature, it is not unreasonable for clients to apply some of the guidelines here even in cases where there is no established relationship between the SIP and the XMPP services (for example, it is reasonable for a client to provide a way for its users to easily start a call to a phone number or SIP URI found in a vCard or obtained from a user directory). However, the strategies to pursue in such cases are left to application developers.

This document makes a further simplifying assumption by discussing only the use of a single client, not use of and coordination among multiple endpoints controlled by the same user (e.g., user agents running simultaneously on a laptop computer, tablet, and mobile phone). Although user agents running on separate endpoints might themselves be CUSAX clients or might engage in different aspects of an interaction (e.g., a user might employ her mobile phone for audio and her tablet for video and text chat), such usage complicates the guidelines for developers of user agents and therefore is left as a matter of implementation for now.

It is important to note that this document does not attempt to standardize "best current practices" in the sense defined in the Internet Standards Process [RFC2026]. Instead, it collects together informational documentation about some strategies that might prove helpful to those who implement and deploy combined SIP-XMPP software and services. With sufficient use and appropriate modification to incorporate the lessons of experience, these strategies might someday form the basis for standardization of best current practices.

2. Client Bootstrap

One of the main problems of using two distinct protocols when providing one service is the impact on usability. Email services, for example, have long been affected by the mixed use of SMTP for outgoing mail and POP3 or IMAP for incoming mail. Although standard service discovery methods (such as the proper DNS records) make it possible for a user agent to locate the right host(s) for connect purposes, they do not provide the kind of detailed information that is needed to actually configure the user agent for use with the service. As a result, it is rather complicated for inexperienced users to configure a mail client and start using it with a new service, and as a result Internet service providers often need to provide configuration instructions for various mail clients. Client developers and communication device manufacturers on the other hand often ship with a number of so-called "wizard" interfaces that enable users to easily configure accounts with a number of popular email services. Although this may improve the situation to some extent, the user experience is still clearly sub-optimal.

While it should be possible for CUSAX users to manually configure their separate SIP and XMPP accounts (often using "wizards"), service providers offering CUSAX services to users of dual-stack SIP/XMPP clients ought to provide methods for online provisioning, typically by means of a web-based service at an HTTPS URL (naturally, single-purpose SIP services or XMPP services could offer such methods as well, but they can be especially helpful where the two aspects of the CUSAX service need to have several configuration options in common).

Although the specifics of such mechanisms are outside the scope of this document, they should make it possible for a service provider to remotely configure the clients based on minimal user input (e.g., only a user ID and password). As far as the authors are aware, no open protocol for endpoint configuration is yet available and adopted; however, application developers are encouraged to explore the potential for future progress in this space (e.g., perhaps based on technologies such as WebFinger [RFC7033]).

By default, when a CUSAX client is used in concert with SIP and XMPP accounts that have a CUSAX relationship (see Section 3.4), the client should disable audio and video calling over XMPP and disable instant messaging and presence over SIP. (It is a matter of implementation whether a CUSAX client allows a user to override these defaults in various ways, e.g., by domain, by individual contact, or by device.) The main advantage of this approach is that a client would employ the most relevant features from both SIP and XMPP when used in the context of a CUSAX service. Note that this default configuration does not apply to standalone SIP accounts or XMPP accounts, for which other settings are likely to be more appropriate (see Section 3.4 for details).

Once a client has been provisioned, it needs to independently log into the SIP account and XMPP account that make up the CUSAX "service" and then maintain both connections.

In order to improve the user experience, when reporting connection status a CUSAX client may wish to present the XMPP connection as an "instant messaging" or a "chat" account and the SIP connection as a "Voice and Video" or a "Telephony" connection. The exact naming is of course entirely up to implementers. The point is that, in cases where SIP and XMPP are components of a service offered by a single provider, such presentation could help users better understand why they are being shown two different connections for what they perceive as a single service. It could alleviate especially situations where one of these connections is disrupted while the other one is still active. Alternatively, the developers of a CUSAX client or the providers of a CUSAX service might decide to force a client to completely disconnect unless both aspects are successfully connected.

Clients may also choose to delay their XMPP connection until they have been successfully registered on SIP. This would help avoid the situation where a user appears online to her contacts but calling the user's client would fail because the user's client is still connecting to the SIP aspect of the CUSAX service.

3. Operation

Once a CUSAX client has been provisioned and authorized to connect to the corresponding SIP and XMPP services it would proceed by retrieving its XMPP roster.

The client should use XMPP for most forms of communication with the contacts from this roster, which will occur naturally because they were retrieved through XMPP. Audio/video features however, would typically be disabled in the XMPP stack, so media-related communication based on these features (e.g., direct calls, conferences, desktop streaming, etc.) would happen over SIP. The rest of this section describes deployment, discovery, usability and linking semantics that enable CUSAX clients to seamlessly use SIP for these features.

3.1. Server-Side Setup

In order for CUSAX to function properly, XMPP service administrators should make sure that at least one of the vCard [RFC6350] "tel" fields for each contact is properly populated with a SIP URI for the user's address at the SIP audio/video service provided by the CUSAX server. There are no limitations as to the form of that number. For example while it is desirable to maintain a certain consistency between SIP AORs and XMPP JIDs, that is by no means required. It is quite important however that the phone number or SIP AOR stored in the vCard be reachable through the SIP aspect of this CUSAX service. (The same considerations apply even if the directory storage format is not vCard storage over XMPP as described by [XEP-0054] or [XEP-0292].)

Administrators may also choose to include the "video" tel type defined in [RFC6350] for accounts that would be capable of handling video communication.

To ensure that the foregoing approach is always respected, service providers might consider validating the values of vCard "tel" fields before storing changes. Of course such validation would be feasible only in cases where a single provider controls both the XMPP and the SIP service since such providers would "know" (e.g., based on use of a common user database for both services) what SIP AOR corresponds to a given XMPP user.

3.2. Service Management

The task of operating and managing a standalone SIP service or XMPP service is not always easy. Combining the two into a unified service introduces additional challenges, including:

- o The necessity of opening additional ports on the client side if SIP functionality is added to an existing XMPP deployment or vice-versa.
- o The potential for important differences in security posture across SIP and XMPP (e.g., SIP servers and XMPP servers might support different TLS ciphersuites).
- o The need for, ideally, a common authentication backend and other infrastructure that is shared across the SIP and XMPP aspects of the combined service.
- o Coordinated monitoring and logging of the SIP and XMPP servers to enable the correlation of incidents and the pinpointing of problems.
- o The difficulty of troubleshooting client-side issues, e.g. if the client loses connectivity for XMPP but maintains its SIP connection.

Although separation of functionality (SIP for media, XMPP for IM and presence) can help to ease the operational burden to some extent, service providers are urged to address the foregoing challenges and similar issues when preparing to launch a CUSAX service.

Beyond the issues listed above, service providers might want to be aware of more subtle operational issues that can arise. For example, if a service provider uses different network operators for the SIP service and the XMPP service, end-to-end connectivity might be more reliable or consistent in one service than in the other service. Similar issues can arise when the media path and the signaling path go over different networks, even in standalone SIP or XMPP services. Providers of CUSAX services are advised to consider the potential for such topologies to cause operational challenges.

3.3. Client-Side Discovery and Usability

When rendering the roster for a particular XMPP account CUSAX clients should make sure that users are presented with a "Call" option for each roster entry that has a properly set "tel" field. This is the case even if calling features have been disabled for that particular XMPP account, as advised by this document. The usefulness of such a feature is not limited to CUSAX. After all, numbers are entered in vCards or stored in directories in order to be dialed and called. Hence, as long as an XMPP client has any means of conducting a call it may wish to make it possible for the user to easily dial any numbers that it learned through whatever means.

Clients that have separate triggers (e.g., buttons) for audio calls and video calls may choose to use the presence or absence of the "video" tel type defined in [RFC6350] as the basis for choosing whether to enable or disable the possibility for starting video calls (i.e., if there is no "video" tel type for a particular contact, the client could disable the "video call" button for that contact).

In addition to discovering phone numbers from vCards or user directories, clients may also check for alternative communication methods as advertised in XMPP presence broadcasts and Personal Eventing Protocol nodes as described in XEP-0152: Reachability Addresses [XEP-0152]. However, these indications are merely hints, and a receiving client ought not associate a SIP address and an XMPP address unless it has some way to verify the relationship (e.g., the vCard of the XMPP account lists the SIP address and the vCard of the SIP account lists the XMPP address, or the relationship is made explicit in a record provided by a trusted directory). Alternatively or in cases where vCard or directory data is not available, a CUSAX client could take the user's own address book as the canonical source for contact addresses.

3.4. Indicating a Relationship Between SIP and XMPP Accounts

In order to improve usability, in cases where clients are provisioned with only a single telephony-capable account they ought to initiate calls immediately upon user request without asking users to indicate an account that the call should go through. This way CUSAX users (whose only account with calling capabilities is usually the SIP part of their service) would have a better experience, since from the user's perspective calls "just work at the click of a button".

In some cases however, clients will be configured with more than the two XMPP and SIP accounts provisioned by the CUSAX provider. Users are likely to add additional stand-alone XMPP or SIP accounts (or accounts for other communications protocols), any of which might have both telephony and instant messaging capabilities. Such situations can introduce additional ambiguity since all of the telephony-capable accounts could be used for calling the numbers the client has learned from vCards or directories.

To avoid such confusion, client implementers and CUSAX service providers may choose to indicate the existence of a special relationship between the SIP and XMPP accounts of a CUSAX service. For example, let's say that Alice's service provider has opened both an XMPP account and a SIP account for her. During or after provisioning, her client could indicate that `alice@xmpp.example.com` has a CUSAX relationship to `alice@sip.example.com` (i.e., that they are two aspects of the same service). This way whenever Alice

triggers a call to a contact in her XMPP roster, the client would preferentially initiate this call through her example.com SIP account even if other possibilities exist (such as the XMPP account where the vCard was obtained or a SIP account with another provider). Similarly, the client would preferentially initiate textual chat sessions using her XMPP account.

If, on the other hand, no relationship has been configured or discovered between a SIP account and an XMPP account, and the client is aware of multiple telephony-capable accounts, it ought to present the user with the option of using XMPP Jingle as one method for engaging in audio and video interactions with a contact who has an XMPP address. This can help to ensure that a CUSAX user can complete audio and video calls with XMPP users who are not part of a CUSAX deployment.

3.5. Matching Incoming SIP Calls to XMPP JIDs

When receiving a SIP call, a CUSAX client may wish to determine the identity of the caller and a corresponding XMPP roster entry so that the receiving user could revert to chatting or other forms of communication that require XMPP. To do so, a CUSAX client could search the user's roster for an entry whose vCard has a "tel" field matching the originator of the call. In addition, in order to avoid the effort of iterating over the entire roster of the user and retrieving vCards for all of the user's contacts, the receiving client may guess at the identity of the caller based a SIP Call-Info header whose 'purpose' header field parameter has a value of "impp" as described in [RFC6993]. To enable this usage, a sending client would need to include such a Call-Info header in the SIP messages that it sends when initiating a call. An example follows.

```
Call-Info: <xmpp:alice@xmpp.example.com> ;purpose=impp
```

Note that the information from the Call-Info header should only be used as a cue: the actual AOR-to-JID binding would still need to be confirmed by the vCard of a contact in the receiving user's roster or through some other trusted means (such as an enterprise directory). If this confirmation succeeds the client would not need to search the entire roster and retrieve all vCards. Not performing the check might enable any caller (including malicious ones) to employ someone else's identity and perform various scams or Man-in-the-Middle attacks.

However, although an AOR-to-JID binding can be a helpful hint to the user, nothing in the foregoing paragraph ought to be construed as necessarily discouraging users, clients, or service providers from

accepting calls originated by entities that are not established contacts of the user (e.g., as reflected in the user's roster); that is a policy matter for the user, client, or service provider.

4. Multi-Party Interactions

CUSAX clients that support the SIP conferencing framework [RFC4353] can detect when a call they are participating in is actually a conference and can then subscribe to conference state updates as per [RFC4575]. A regular SIP user agent might also use the same conference URI for text communication with the Message Session Relay Protocol (MSRP). However, given that SIP's instant messaging capabilities would normally be disabled (or simply not supported) in CUSAX deployments, an XMPP Multi-User Chat (MUC) room [XEP-0045] associated with the conference can be announced/discovered through <service-uris> bearing the "groupextchat" purpose [I-D.ivov-groupextchat-purpose]. Similarly, an XMPP MUC room can advertise the SIP URI of an associated service for audio/video interactions using the 'audio-video-uri' field of the "muc#roominfo" data form [XEP-0004] to include extended information [XEP-0128] about the MUC room within XMPP service discovery [XEP-0030]; see [XEP-0045] for an example. These methods would enable a CUSAX-aware SIP conference server to advertise the existence of an associated XMPP chatroom, and for a CUSAX-aware XMPP chatroom to advertise the existence of an associated SIP conference server.

If a CUSAX client joins the MUC room associated with a particular call, it should not rely on any synchronization between the two. Both the SIP conference and the XMPP MUC room would function independently, each issuing and delivering its own state updates. Hence it is possible that that certain peers would temporarily or permanently be reachable in only one of the two conferences. This would typically be the case with single-stack clients that have only joined the SIP call or the XMPP MUC room. It is therefore important for CUSAX clients to provide a clear indication to users as to the level of involvement of the various participants: i.e., a user needs to be able to easily understand whether a certain participant can receive text messages, audio/video, or both.

At the level of the CUSAX service, it is also possible to enforce tighter integration between the XMPP MUC room and the SIP conference. Permissions, roles, kicks and bans that are granted and performed in the MUC room can easily be imitated by the conference focus/mixer into the SIP call. If, for example, a certain MUC member is muted, the conference mixer can choose to also apply the mute on the media stream corresponding to that participant. However, the details and exact level of such integration are entirely up to implementers and service providers.

The approach above describes one relatively lightweight possibility of combining SIP and XMPP multi-party interaction semantics without requiring tight integration between the two. As with the rest of this document, this approach is by no means normative. Implementations and future documents may define other methods or provide other suggestions for improving the unified communications user experience in cases of multi-user chats and conference calling.

5. Federation

In theory there are no technical reasons why federation (i.e., inter-domain communication) would require special behavior from CUSAX clients. However, it is worth noting that differences in administration policies may sometimes lead to potentially confusing user experiences.

For example, let's say atlanta.example.com observes the CUSAX policies described in this document. All XMPP users at atlanta.example.com are hence configured to have vCards that match their SIP identities. Alice is therefore used to making free, high-quality SIP calls to all the people in her roster. Alice can also make calls to the PSTN by simply dialing numbers. She may even be used to these calls being billed to her online account so she would be careful about how long they last. This is not a problem for her since she can easily distinguish between a free SIP call (one that she made by calling one her roster entries) from a paid PSTN call that she dialed as a number.

Then Alice adds xmpp:bob@biloxi.example.com. The Biloxi domain only has an XMPP service. There is no SIP server and Bob uses an XMPP-only client. However, Bob has added his mobile number to his vCard in order to make it easily accessible to his contacts. Alice's client would pick up this number and make it possible for Alice to start a call to Bob's mobile phone number.

This could be a problem because, other than the fact that Bob's address is from a different domain, Alice would have no obvious and straightforward cues telling her that this is in fact a call to the PSTN. In addition to the potentially lower audio quality, Alice may also end up incurring unexpected charges for such calls.

In order to avoid such issues, providers maintaining a CUSAX service for the users in their domain may choose to provide additional cues (e.g., a service-generated signal that triggers a user interface warning in a CUSAX client, an auditory tone, or a spoken message) indicating that a call would incur unexpected charges.

Another scenario arises when a SIP service allow communication only with intra-domain numbers; here Alice might be prevented from establishing a call with Bob's mobile phone. Providers should therefore make sure that calls to inter-domain numbers are flagged with an appropriate audio or textual warning.

6. Summary of Suggested Strategies

The following strategies are suggested for CUSAX user agents:

1. By default, prefer SIP for audio and video, and XMPP for messaging and presence.
2. Use XMPP for all forms of communication with the contacts from the XMPP roster, with the exception of features that are based on establishing real-time sessions (e.g. audio/video calls), for which SIP should be used.
3. Provide online provisioning options for providers to remotely setup SIP and XMPP accounts so that users wouldn't need to go through a multi-step configuration process.
4. Provide online provisioning options for providers to completely disable features for an account associated with a given protocol (SIP or XMPP) if the features are preferred in another protocol (XMPP or SIP).
5. Present a "Call" option for each roster entry that has a properly set "tel" field in the vCard or equivalent.
6. If the client is provisioned with only a single telephony-capable account, initiate calls immediately upon user request without asking users to indicate an account that the call should go through.
7. If no relationship has been configured or discovered between a SIP account and an XMPP account, and the client is aware of multiple telephony-capable accounts, present the user with the choice of reaching the contact through any of those accounts.
8. If known, indicate the existence of a special relationship between the SIP and XMPP accounts of a CUSAX service.
9. Optionally, present the XMPP connection as an "instant messaging" or a "chat" account and the SIP connection as a "Voice and Video" or a "Telephony" account.

10. Optionally, determine the identity of the audio/video caller and a corresponding XMPP roster entry so that the user could use textual chatting or other forms of communication that require XMPP.
11. Optionally, delay the XMPP connection until after a SIP connection has been successfully registered.
12. Optionally, check for alternative communication methods (SIP addresses advertised over XMPP, and XMPP addresses advertised over SIP).

The following strategies are suggested for CUSAX services:

1. Use online provisioning and configuration of accounts so that users won't need to setup two separate accounts for the CUSAX service.
2. Use online provisioning so that calling features are disabled for all XMPP accounts.
3. Ensure that at least one of the vCard "tel" fields for each XMPP user is properly populated with a SIP URI that is reachable through the SIP service.
4. Optionally, include the "video" tel type for accounts that are capable of handling video communication.
5. Optionally, provision clients with information indicating that specific SIP and XMPP accounts are related in a CUSAX service.
6. Optionally, attach a "Call-Info" header with an "impp" purpose to all SIP INVITE messages, so that clients can more rapidly associate a caller with a roster entry and display a "Caller ID".

7. IANA Considerations

This document has no actions for the IANA.

8. Security Considerations

Use of the same user agent with two different accounts providing complementary features introduces the possibility of mismatches between the security profiles of those accounts or features. Two security mismatches of particular concern are:

- o The SIP aspect and XMPP aspect of a CUSAX service might offer different authentication options (e.g., digest authentication for

SIP as specified in [RFC3261] and SCRAM authentication [RFC5802] for XMPP as specified in [RFC6120]). Because SIP uses a password-based method (digest) and XMPP uses a pluggable framework for authentication via the Simple Authentication and Security Layer (SASL) technology [RFC4422], it is also possible that the XMPP connection could be authenticated using a password-free method such as client certificates with SASL EXTERNAL even though a username and password is used for the SIP connection.

- o The Transport Layer Security (TLS) [RFC5246] ciphersuites offered or negotiated on the XMPP side might be different from those on the SIP side because of implementation or configuration differences between the SIP server and the XMPP server. Even more seriously, a CUSAX client might successfully negotiate TLS when connecting to the XMPP aspect of the service but not when connecting to the SIP aspect, or vice-versa. In this situation an end user might think that the combined CUSAX session with the service is protected by TLS, even though only one aspect is protected.

Security mismatches such as these (as well as others related to end-to-end encryption of messages or media) introduce the possibility of downgrade attacks, eavesdropping, information leakage, and other security vulnerabilities. User agent developers and service providers must ensure that such mismatches are avoided as much as possible (e.g., by enforcing common and strong security configurations and policies across protocols). Specifically, if both protocols are not safeguarded by similar levels of cryptographic protection, the user must be informed of that fact and given the opportunity to bring both up to the same level.

Section 5 discusses potential issues that may arise due to a mismatch between client capabilities, such as calls being initiated with costs that are not expected by the end user. Such issues could be triggered maliciously, as well as by accident. Implementers therefore need to provide necessary cues to raise user awareness as suggested in Section 5.

Refer to the specifications for the relevant SIP and XMPP features for detailed security considerations applying to each "stack" in a CUSAX client.

9. References

9.1. Normative References

- [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.

[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.

9.2. Informative References

[I-D.ivov-grouptextchat-purpose]

Ivov, E., "A Group Text Chat Purpose for Conference and Service URIs in the Session Initiation Protocol (SIP) Event Package for Conference State ", draft-ivov-grouptextchat-purpose-03 (work in progress), June 2013.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, February 2006.

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5802] Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams, "Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms", RFC 5802, July 2010.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, August 2011.

[RFC6914] Rosenberg, J., "SIMPLE Made Simple: An Overview of the IETF Specifications for Instant Messaging and Presence Using the Session Initiation Protocol (SIP)", RFC 6914, April 2013.

- [RFC6993] Saint-Andre, P., "Instant Messaging and Presence Purpose for the Call-Info Header Field in the Session Initiation Protocol (SIP)", RFC 6993, July 2013.
- [RFC7033] Jones, P., Salgueiro, G., Jones, M., and J. Smarr, "WebFinger", RFC 7033, September 2013.
- [XEP-0004] Eatmon, R., Hildebrand, J., Miller, J., Muldowney, T., and P. Saint-Andre, "Data Forms", XSF XEP 0004, August 2007.
- [XEP-0030] Hildebrand, J., Millard, P., Eatmon, R., and P. Saint-Andre, "Service Discovery", XSF XEP 0030, June 2008.
- [XEP-0045] Saint-Andre, P., "Multi-User Chat", XSF XEP 0045, February 2012.
- [XEP-0054] Saint-Andre, P., "vcard-temp", XSF XEP 0054, July 2008.
- [XEP-0128] Saint-Andre, P., "Service Discovery Extensions", XSF XEP 0128, October 2004.
- [XEP-0152] Hildebrand, J. and P. Saint-Andre, "XEP-0152: Reachability Addresses", XEP XEP-0152, September 2013.
- [XEP-0166] Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S., and J. Hildebrand, "Jingle", XSF XEP 0166, December 2009.
- [XEP-0167] Ludwig, S., Saint-Andre, P., Egan, S., McQueen, R., and D. Cionoiu, "Jingle RTP Sessions", XSF XEP 0167, December 2009.
- [XEP-0292] Saint-Andre, P. and S. Mizzi, "vCard4 Over XMPP", XSF XEP 0292, September 2013.

Appendix A. Acknowledgements

This draft is inspired by the "SIXPAC" work of Markus Isomaki and Simo Veikkolainen. Markus also provided various suggestions for improving the document.

The authors would also like to thank the following people for their reviews and suggestions: Sebastien Couture, Dan-Christian Bogos, Richard Brady, Olivier Crete, Aaron Evans, Kevin Gallagher, Adrian Georgescu, Saul Ibarra Corretge, David Laban, Gergely Lukacsy, Murray Mar, Daniel Pocock, Travis Reitter, and Gonzalo Salgueiro.

Brian Carpenter, Ted Hardie, Paul Hoffman, and Benson Schliesser reviewed the document on behalf of the General Area Review Team, the Applications Area Directorate, the Security Directorate, and the Operations and Management Directorate, respectively.

Benoit Claise, Barry Leiba, and Pete Resnick provided helpful and substantive feedback during IESG review.

The document shepherd was Mary Barnes. The sponsoring Area Director was Gonzalo Camarillo.

Authors' Addresses

Emil Ivov
Jitsi
Strasbourg 67000
France

Phone: +33-177-624-330
Email: emcho@jitsi.org

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

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

Enrico Marocco
Telecom Italia
Via G. Reiss Romoli, 274
Turin 10148
Italy

Email: enrico.marocco@telecomitalia.it

MMUSIC
Internet-Draft
Intended status: Informational
Expires: January 10, 2014

J. Marcon
R. Ejzak
Alcatel-Lucent
July 09, 2013

MSRP over WebRTC data channels
draft-marcon-msrp-over-webrtc-data-channels-00

Abstract

The Real-Time Communication in WEB-browsers (RTCWeb) working group is charged to provide protocols to support direct interactive rich communication using audio, video, and data between two peers' web-browsers. For the support of data communication, the RTCWeb working group has in particular defined the concept of bi-directional data channels over SCTP, where each data channel might be used to transport other protocols, called sub-protocols. This document specifies how the Message Session Relay Protocol (MSRP) can be instantiated as a WebRTC data channel sub-protocol, using the SDP offer/exchange to negotiate out-of-band the sub-protocol specific parameters. Two network configurations are documented: a WebRTC end-to-end configuration (connecting two MSRP over data channel endpoints), and a gateway configuration (connecting an MSRP over data channel endpoint with an MSRP over TCP endpoint).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 10, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Conventions	3
3. Terminology	3
4. WebRTC Data Channels	4
5. End-to-end configuration	5
5.1. Support for SDP-based sub-protocol negotiation	5
5.1.1. SDP syntax	5
5.1.1.1. Channel-specific setup parameters	5
5.1.1.2. Sub-protocol specific attributes	6
5.1.2. Procedures	7
5.1.2.1. Opening a data channel	7
5.1.2.2. Closing a data channel	7
5.2. Support for MSRP data channels	7
5.2.1. Overview	7
5.2.2. MSRP URI	8
5.2.3. Session negotiation	8
5.2.4. Session opening	8
5.2.5. Data sending and reporting	8
5.2.6. Session closing	9
5.3. Support for MSRP File Transfer function	9
6. Gateway configuration	9
7. Security Considerations	10
8. IANA Considerations	10
9. Acknowledgments	10
10. References	10
10.1. Normative References	10
10.2. Informative References	11
Authors' Addresses	12

1. Introduction

The Message Session Relay Protocol (MSRP) [RFC4975] is currently defined to work over TCP connections.

The RTCWeb working group has defined the concept of bi-directional data channels running on top of SCTP/DTLS. Each data channel consists of paired SCTP streams sharing the same SCTP Stream Identifier. Data channels are created by endpoint applications through the WebRTC API, and can be used to transport proprietary or well-defined protocols, which in the latter case can be signaled by the data channel "sub-protocol" parameter, conceptually similar to the WebSocket "sub-protocol". However, apart from the "sub-protocol" value transmitted to the peer, RTCWeb leaves open how endpoint applications can agree on how to instantiate a given sub-protocol on a data channel, whether it be in-band or out-of-band (or both). As an example, the SDP offer generated by the browser includes no channel-specific information.

MSRP is a protocol for transmitting a series of related instant messages in the context of a session. In addition to instant messaging, MSRP can also be used for image sharing or file transfer.

Defining the MSRP as a data channel sub-protocol has many benefits:

- o provide to WebRTC applications a proven protocol enabling instant messaging, file transfer, image sharing
- o integrate those features with other RTCWeb voice and video features
- o leverage the SDP-based negotiation already defined for MSRP
- o allows the interworking with MSRP endpoints running on a TCP connection

This document defines the use MSRP of over WebRTC data channels, where one MSRP endpoint is an MSRP WebRTC application and the other endpoint is either an MSRP WebRTC application or an MSRP TCP application.

2. Conventions

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

3. Terminology

This document uses the following terms:

Data channel: A bidirectional channel consisting of paired SCTP outbound and inbound streams.

In-band: transmission through the peer-to-peer SCTP association.

Out-of-band: transmission through the WebRTC signaling path, using JSEP [I-D.ietf-rtcweb-jsep] and the SDP Offer/Answer model [RFC3264].

MSRP data channel: A data channel specifically used to transport the messages of one MSRP session.

Peer: From the perspective of one of the agents in a session, its peer is the other agent. Specifically, from the perspective of the SDP offerer, the peer is the SDP answerer. From the perspective of the SDP answerer, the peer is the SDP offerer.

4. WebRTC Data Channels

This section summarizes how WebRTC data channels work in general.

A WebRTC application creates a data channel via the WebRTC Data Channel API, by providing a number of setup parameters (sub-protocol, label, reliability, order of delivery, priority).

The browser then opens in-band the data channel using the DATA CHANNEL OPEN message defined in [draft-jesup-rtcweb-data-protocol]. This message carries some of the channel-specific parameters passed by the application (sub-protocol, label, reliability, order of delivery).

In case an SCTP association is already established, the browser transmits immediately the DATA CHANNEL OPEN message to the peer, on an unused SCTP stream.

In case no SCTP association is established, the browser triggers for an SDP offer/answer exchange, and sends the DATA CHANNEL OPEN message(s) once the SCTP association is established (i.e. subsequently to the reception of the answer).

The SDP offer generated by the browser is as per [draft-ietf-mmusic-sctp-sdp]. In brief, it contains one m-line for the SCTP association on top of which data channels will run, and one attribute per protocol assigned to the SCTP ports:

```
m=application 54111 DTLS/SCTP 5000 5001 5002
c=IN IP4 79.97.215.79
a=sctpmap:5000 webrtc-datachannel 16
```

```
a=sctpmap:5001 bfcf 2
a=sctpmap:5002 t38 1
```

Note: A WebRTC browser will only create an sctpmap attribute for the webrtc-datachannel protocol, and will not create sctpmap attributes for other protocols such as bfcf or t38. This example shows the hypothetical power of the syntax to support multiplexing of SCTP associations for different protocols on the same DTLS connection.

Note: this SDP syntax does not contain any channel-specific information.

5. End-to-end configuration

This section describes the network configuration where each MSRP endpoint is a WebRTC endpoint, running MSRP over an SCTP/DTLS connection.

5.1. Support for SDP-based sub-protocol negotiation

In the default procedures described in Section 4, the channel-specific parameters are notified in-band to the peer, rather than negotiated with the peer. Also, no mechanism is defined to transmit subprotocol-specific parameters to the peer.

This section defines a means to negotiate channel-specific and subprotocol-specific parameters, using the out-of-band SDP offer/exchange.

5.1.1. SDP syntax

The SDP only contains the declaration of data channels for which an SDP-based negotiation is required, and that are either being created or already opened.

5.1.1.1. Channel-specific setup parameters

For each of these data channels, the SDP lists one attribute line providing the Stream Identifier, sub-protocol, label, reliability, order of delivery, priority.

```
a=webrtc-DataChannel:5000 stream=2;label="channel 2"; \
  subprotocol="file transfer";max_retr=3
```

NOTE: the related SDP syntax has to be imported from version 3 of [draft-ietf-mmusic-sctp-sdp].

This line MUST be replicated without changes in the SDP answer, if the answerer accepts the offered data channel.

This line MUST be replicated without changes in any subsequent offer or answer, as long as the data channel is still opened at the time of offer or answer generation.

The Sub-protocol, label, reliability and order of delivery parameters MUST be equal to those transmitted in-band in the DATA CHANNEL OPEN message. The Stream Identifier MUST be equal to the SCTP Stream Identifier on which the DATA CHANNEL OPEN message is sent.

5.1.1.2. Sub-protocol specific attributes

In the SDP, each data channel declaration MAY also be followed by other SDP attributes specific to the sub-protocol in use. Each of these attributes is represented by one new attribute line, and it includes the contents of a media-level SDP attribute already defined for use with this (sub)protocol in another IETF specification.

Each sub-protocol specific attribute such as "a=accept-types:text/plain" that would normally be used to negotiate an instance of MSRP is replaced with an attribute of the form "a=wdcsa:sctp-port:stream-id original-attribute", where wdcsa stands for "webrtc-DataChannel sub-protocol attribute", sctp-port is the sctp port number assigned for webrtc-DataChannel on the media line, stream-id is the sctp stream id assigned to this instance of MSRP, and original-attribute represents the contents of the MSRP related attribute to be included.

```
a=webrtc-DataChannel:5000 stream=2;label="channel 2"; \
  subprotocol="MSRP";max_retr=3
a=wdcsa:5000:2 accept-types:text/plain
```

Thus the attribute "a=wdcsa:5000:2 accept-types:text/plain" specifies that this instance of MSRP on stream id 2 accepts plain text files.

As opposed to the data channel setup parameters, these parameters are subject to offer/answer negotiation.

The same syntax applies to any other SDP attribute required for negotiation of this instance of the sub-protocol.

5.1.2. Procedures

5.1.2.1. Opening a data channel

Opening a data channel is done in-band by the DATA CHANNEL OPEN message. However when the sub-protocol requires an SDP-based negotiation, applications **MUST NOT** send data on this channel till both SDP negotiation and DATA CHANNEL OPEN message sending are done, which may happen in any order.

When the application creates a new data channel (requiring some sub-protocol specific negotiation), the browser follows in any case a generic behavior:

- o if no SCTP association is established, the browser triggers the SDP negotiation, and sends the DATA CHANNEL OPEN message once the answer is received and the SCTP association initialized.
- o if an SCTP association is established, the browser does not trigger any SDP negotiation but instead immediately sends a DATA CHANNEL OPEN message. The application then initiates a new offer/answer exchange

Note: in this case, as the DATA CHANNEL OPEN message is sent before the offer is created, Stream ID conflicts between offers sent to the peer, and DATA CHANNEL OPEN messages received from the peer should not occur.

The application has the task to complete the browser-generated offer (or answer) with the data channel and subprotocol specific parameters in scope of the SCTP m-line. The browser is expected to ignore those parameters when the completed offer (or answer) is applied locally.

5.1.2.2. Closing a data channel

Closing a data channel is done in-band by the SSN reset mechanism, and does not trigger a new offer/answer exchange.

5.2. Support for MSRP data channels

5.2.1. Overview

This document defines how MSRP can be used as a WebRTC sub-protocol, where the MSRP-related negotiation is done as part of the SDP-based data channel negotiation defined in Section 5.1.1.2.

In this design, the MSRP connection maps to the SCTP association and the port assigned to data channels, and each MSRP session maps to one data channel exactly.

5.2.2. MSRP URI

This document extends the MSRP URI syntax [RFC3986] by defining the new transport parameter value "dc":

```
transport = "tcp" / "dc" / 1*ALPHANUM
```

5.2.3. Session negotiation

Using the syntax `a=webrtc-DataChannel:<port> <param=value>`, the SDP declaration of a given MSRP data channel can include at least all the following well-known parameters:

- o defined in [RFC4975]: "path", "accept-types", "accept-wrapped-types", "max-size"
- o defined in [RFC4566]: "sendonly", "recvonly", "inactive", and "sendrecv"
- o defined in [RFC6135]: "setup"
- o defined in [RFC6714]: "msrp-cema"
- o defined in [RFC5547]: all the parameters related to MSRP file transfer

5.2.4. Session opening

The MSRP session is normally opened by the active MSRP endpoint, which sends an MSRP SEND message (empty or not) to the other MSRP endpoint. The active MSRP endpoint does not use the path attribute to open a transport connection to its peer. Instead, the active MSRP endpoint uses the DataChannel established for this MSRP session by the procedures in Section 5.1. The cema attribute is implicitly associated with every MSRP session using data channel transport.

5.2.5. Data sending and reporting

5.2.6. Session closing

Either endpoint can close the MSRP session by closing the underlying data channel. Closing an MSRP session should not trigger an SDP negotiation.

5.3. Support for MSRP File Transfer function

[RFC5547] defines an end-to-end file transfer method based on MSRP and the SDP offer/answer mechanism. This file transfer method is also usable by MSRP WebRTC endpoints, with the following considerations:

- o As an MSRP session maps to one data channel, a file transfer session maps also to one data channel.
- o SDP attributes specified in [RFC5547] for a file transfer m-line are embedded as subprotocol-specific attributes as defined in Section 5.1.1.2.
- o Each file chunk is transmitted over one SCTP user message.
- o Once the file transfer is complete, the same data channel MAY be reused for another file transfer.
- o Following the aborting of a file transfer, the SDP can be updated by adding the "inactive" attribute to the list of subprotocol-specific attributes associated with the corresponding data channel.

6. Gateway configuration

This section describes the network configuration where one endpoint runs MSRP over a WebRTC SCTP/DTLS connection, the other MSRP endpoint runs MSRP over one or more TLS/TCP connections, and the two endpoints interwork via an MSRP gateway.

Specifically, a gateway can be configured to interwork an MSRP session using a data channel with a peer that does not support data channel transport in one of two ways. In one model, the gateway performs as a MSRP B2BUA to interwork all the procedures as necessary between the endpoints. No further specification is needed for this model.

Alternately, the gateway can use CEFA procedures to provide transport level interworking between MSRP endpoints using different transport protocols as follows.

When the gateway performs transport level interworking between MSRP endpoints, all of the procedures in section Section 5.1 apply to each peer, with the following additions:

- o The endpoint establishing an MSRP session using data channel transport shall not request inclusion of any relays, although it may interoperate with a peer that signals the use of relays.
- o The gateway receiving an SDP offer that includes a request to negotiate an MSRP session on a data channel can provide transport level interworking in the same manner as a CEMA SBC by forwarding TCP or TLS transport parameters in a new m line with the appropriate attributes within the forwarded SDP offer.
- o Similarly, a gateway receiving an SDP offer to negotiate an MSRP session using TCP or TLS transport with an endpoint that only supports data channel transport for MSRP can provide transport level interworking in the same manner as a CEMA SBC by establishing a new data channel for the MSRP session with the target endpoint.

7. Security Considerations

To be completed.

8. IANA Considerations

To be completed.

9. Acknowledgments

The authors wish to thank... for their invaluable comments.

10. References

10.1. Normative References

- [I-D.ietf-rtcweb-jsep]
Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-02 (work in progress), October 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

- [RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.
- [RFC4976] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions for the Message Sessions Relay Protocol (MSRP)", RFC 4976, September 2007.
- [RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009.
- [RFC6135] Holmberg, C. and S. Blau, "An Alternative Connection Model for the Message Session Relay Protocol (MSRP)", RFC 6135, February 2011.
- [RFC6714] Holmberg, C., Blau, S., and E. Burger, "Connection Establishment for Media Anchoring (CEMA) for the Message Session Relay Protocol (MSRP)", RFC 6714, August 2012.

10.2. Informative References

- [I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "RTCWeb Data Channels", draft-ietf-rtcweb-data-channel-04 (work in progress), February 2013.
- [I-D.jesup-rtcweb-data-protocol] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Protocol", draft-jesup-rtcweb-data-protocol-04 (work in progress), February 2013.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [WebRtcAPI] Bergkvist, A., Burnett, D., Narayanan, A., and C. Jennings, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD WD-webrtc-20120821, August 2012, <<http://www.w3.org/TR/2012/WD-webrtc-20120821>>.

Authors' Addresses

Jerome Marcon
Alcatel-Lucent
Route de Villejust
Nozay 91620
France

Email: jerome.marcon@alcatel-lucent.com

Richard Ejzak
Alcatel-Lucent
1960 Lucent Lane
Naperville, Illinois 60563-1594
US

Phone: +1 630 979 7036

Email: richard.ejzak@alcatel-lucent.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 18, 2013

P. Saint-Andre
Cisco Systems, Inc.
S. Loreto
Ericsson
F. Forno
Bluendo srl
October 15, 2012

Interworking between the Session Initiation Protocol (SIP) and the
Extensible Messaging and Presence Protocol (XMPP): Multi-Party Text Chat
draft-saintandre-sip-xmpp-groupchat-02

Abstract

This document defines a bi-directional protocol mapping for the exchange of instant messages in the context of a many-to-many chat session among users of the Session Initiation Protocol (SIP) and users 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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
1.1. Overview	4
1.2. Terminology	4
1.3. Scope	4
1.4. Formal and Informal Sessions	5
1.5. Gateway Heuristics	5
1.6. Acknowledgements	5
1.7. Discussion Venue	5
2. XMPP Group Chat to MSRP Multiparty Instant Message (IM) Session	5
2.1. Entering a Room	7
2.2. Setting up a nickname	9
2.3. Presence Broadcast	10
2.4. Exchanging Messages	12
2.4.1. Sending a Message to All Occupants	13
2.4.2. Sending a Private Message	14
2.5. Exiting a Room	14
2.6. Nickname Conflict	16
2.7. Changing Nickname	17
3. MSRP Multiparty Instant Message (IM) Session to XMPP Group Chat	17
3.1. Entering a Room	19
3.2. Presence Broadcast	21
3.3. Exchanging Messages	22
3.3.1. Sending a Message to All Occupants	23
3.3.2. Sending a Private Message	24
3.4. Exiting a Room	25
3.5. Nickname Conflict	25
3.6. Changing Nickname	26
4. Security Considerations	26
5. IANA Considerations	26
6. References	27
6.1. Normative References	27
6.2. Informative References	27
Authors' Addresses	28

1. Introduction

1.1. Overview

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

The architectural assumptions underlying such protocol mappings are provided in [I-D.saintandre-sip-xmpp-core], including mapping of addresses and error conditions. Mappings for single instant messages (sometimes called "pager-mode" messaging) are provided in [I-D.saintandre-sip-xmpp-im]. Mappings for one-to-one text chat sessions are provided in [I-D.saintandre-sip-xmpp-chat].

This document specifies mappings for many-to-many text chat sessions (sometimes called "groupchat"); in particular, this document specifies mappings between XMPP and the Message Session Relay Protocol [RFC4975].

1.2. 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].

1.3. Scope

Both XMPP and SIP/SIMPLE technologies enable multi-user text chat, whereby users can exchange messages in the context of a room. The term "room" usually is a synonym for a virtual environment where people enter and exchange messages.

Groupchat messages are messages which are sent from a sender to multiple recipients (i.e., two or more) in the context of a "multi-user chat session", "text conference", or "chatroom". In XMPP a groupchat message is a <message/> stanza of type "groupchat" that is reflected from the sender to multiple recipients by a multi-user chat service, as defined in [XEP-0045]. In SIP/SIMPLE a groupchat message is reflected from the sender to multiple recipients by a conference server that uses MSRP to handle groupchat sessions, as defined in [I-D.ietf-simple-chat].

As in [I-D.saintandre-sip-xmpp-im] and related 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 "groupchat" are specified in [XEP-0045].
- o SIP-based chat room sessions using the SIP INVITE and SEND request types are specified in [I-D.ietf-simple-chat].

1.4. Formal and Informal Sessions

TBD: Does XMPP use Formal and Informal session also for group-chat?

1.5. Gateway Heuristics

TBD

1.6. Acknowledgements

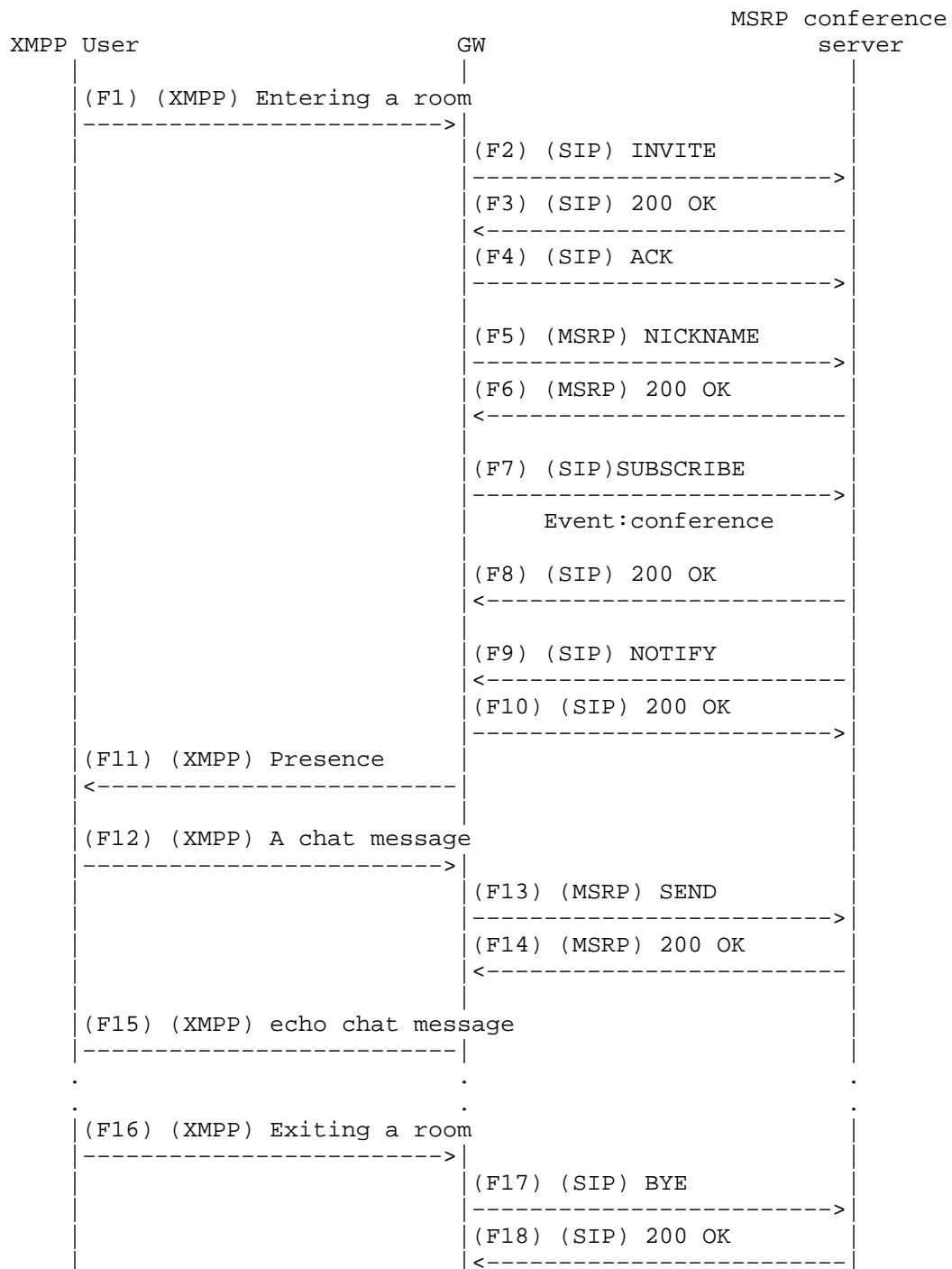
Some text in this document was borrowed from [I-D.saintandre-sip-xmpp-core] and from [XEP-0045].

1.7. Discussion Venue

The authors welcome discussion and comments related to the topics presented in this document. The preferred forum is the <sip-xmpp@xmpp.org> mailing list, for which archives and subscription information are available at <<http://mail.jabber.org/mailman/listinfo/sip-xmpp>>.

2. XMPP Group Chat to MSRP Multiparty Instant Message (IM) Session

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



2.1. Entering a Room

When the XMPP user ("Juliet") wants to join a multi-user chat room ("Verona"), she sends a <presence/> stanza to the hostname hosting that chat room, she also specifies the "nick" she desires to use within the room ("juliet"). The Room Nickname is the resource identifier portion of a Room JID. The Juliet client SHOULD signal 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 entering a chatroom

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

Upon receiving such a presence stanza, the XMPP server to which Juliet has authenticated attempts to deliver the stanza to a local domain or attempts to route the presence stanza to the remote domain that services the hostname in the 'to' attribute. Naturally, in this document we assume that the hostname in the 'to' attribute is an Chat Room-aware SIP 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 "_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.shakespeare.net", since we have already assumed that the shakespeare.net hostname is running a SIP instant messaging service. (Note: The XMPP server may 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 (x2s.example.com), which transforms the presence stanza into SIP syntax and routes it to the remote conference server (shakespeare.net).

As a compliant multi-user chat services MUST accept the presence

stanza containing an empty <x/> element qualified by the 'http://jabber.org/protocol/muc' namespace as a request to enter a room; the XMPP-to-SIP gateway MUST transform it in a SIP INVITE request.

Example: (F2) Juliet entering a chatroom (SIP transformation)

```
INVITE sip:verona@chat.shakespeare.net SIP/2.0
To: <sip:verona@chat.shakespeare.net>
From: <sip:juliet@example.com>;tag=786
Call-ID: 711609sa
Content-Type: application/sdp
Content-Length: [length]

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

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.saintandre-sip-xmpp-core], the mapping of XMPP syntax elements to SIP and [RFC4566] syntax elements SHOULD be as shown in the following table. (Mappings for elements not mentioned are undefined.)

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

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

Here we assume that the chat room server accepts the session establishment. It includes the 'isfocus' and other relevant feature tags in the Contact header field of the response. The chat room

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) the chat room accepts the session establishment

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

c=IN IP4 shakespeare.net
m=message 12763 TCP/MSRP *
a=accept-types:message/cpim
a=accept-wrapped-types:text/plain text/html *
a=path:msrp://s2x.shakespeare.net:12763/kjhd37s2s20w2a;tcp
```

Upon receiving such a response, the SIMPLE server or associated SIP-to-XMPP gateway MUST send a SIP ACK to the SIP user.

Example: (F4) the Gateway sends ACK to the chat room server

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

2.2. Setting up a 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. The nickname is set up using the extension specified in [I-D.ietf-simple-chat]

Example: (F5) the Gateway set up the nickname

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

The chat room server analyzes the existing allocation of nicknames,

accepts the nick name proposal and answers with a 200 response.

Example: (F6) the chat room accepts the nickname proposal

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

2.3. Presence Broadcast

If the multi-user chat service accepts the request to enter a room, the xmpp user expects to receive back presence information from all the existing occupants' 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 send back to the XMPP-to-SIP gateway a NOTIFY with the presence information from all the existing occupants' room

Example: (F9) the chat room notifies the presence information

```
NOTIFY sip:verona@chat.shakespeare.net SIP/2.0
To: Juliet <sip:juliet@example.com>;tag=43524545
From: <sip:verona@chat.shakespeare.net>;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@conf.example.com">
  <conference-description>
    <subject>Today in Verona</subject>
    <conf-uris>
      <entry>
        <uri>tel:+18882934234</uri>
      </entry>
    </conf-uris>
  </conference-description>
  <users>
    <user entity="sip:romeo@example.com" state="full">
      <nickname-text>romeo</nickname-text>
      <roles>
        <entry>participant</entry>
      </roles>
    </user>
  </users>
</conference-info>
```

[NOTE: 1] a full mapping of RFC 4575 will be defined later on.

[NOTE: 2] the <nickname-text/> attribute is an extension to the conference package explained but not defined in [I-D.ietf-simple-chat]

[NOTE: 3] the subject (if present in the NOTIFY) must be sent with a separate <message/> stanza; so after F11 there should be another <message/> stanza from the gw to the joining party

[OPEN ISSUE: 1] how to send to the room jid with the subject child set: do we need to send it in a different presence stanza than the F11?

Upon receiving such a response, the SIP-to-XMPP gateway MUST send a 200 OK to the MSRP conference server and translate it in an xmpp presence stanza.

Example: (F11) the chat room presence information translated in XMPP

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

As specified in ???, the mapping of SIP and SDP syntax elements to XMPP syntax elements SHOULD be as shown in the following table. (Mappings for elements not mentioned are undefined.)

Table 2: Message syntax mapping from SIP/SDP to XMPP

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

[OPEN ISSUE: 1] how to match the <roles/> SIP Conference attribute in the XMPP <affiliation/> and <role/>. In XMPP roles are current privileges within the room while, affiliations are kept permanently in different sessions (they are the default for a given user).

2.4. Exchanging 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 XMPP syntax elements to MSRP syntax elements SHOULD be 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 from <body/>	To From body of the SEND request

2.4.1. Sending a Message to All Occupants

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

The following examples show an exchange of a public message.

Example: (F12) Juliet sends a Message to all occupants

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

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

Example: (F13) Gateway transforms XMPP message to MSRP

```
MSRP a786hjs2 SEND
To-Path: msrp://s2x.shakespeare.net: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.shakespeare.net;transport=tcp>
From: <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, 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.shakespeare.net:12763/kjhd37s2s20w2a;tcp
-----d93kswow$
```

Since the XMPP room could be moderated and an XMPP User can not be sure whether his message has been accepted or not, without an echo

from the server, the [XEP-0045] states that the sender have to receive back the same message it has generated. So in this scenario the XMPP-to-SIP gateway has to generate the echo message.

2.4.2. Sending a Private Message

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

The following examples show an exchange of a private message.

Example: (F12) Juliet sends a private message

```
<message from='juliet@example.com'
      to='verona@chat.shakespeare.net/romeo'
      type='chat' />
  <body>O Romeo, Romeo! wherefore art thou Romeo?</body>
</message>
```

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

Example: (F13) Gateway transforms XMPP message to MSRP

```
MSRP a786hjs2 SEND
To-Path: msrp://s2x.shakespeare.net: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@chat.shakespeare.net>
From: <sip:juliet@chat.shakespeare.net>
DateTime: 2008-10-15T15:02:31-03:00
Content-Type: text/plain

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

2.5. Exiting a Room

If Juliet decides to exit the multi-user chat room, her client sends a presence stanza of type "unavailable" to the <verona@chat.shakespeare.net/juliet> she is currently using in the room.

Example: (F16) Juliet exiting a chatroom

```
<presence from='juliet@example.com'
          to='verona@chat.shakespeare.net/juliet'
          type='unavailable'>
</presence>
```

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

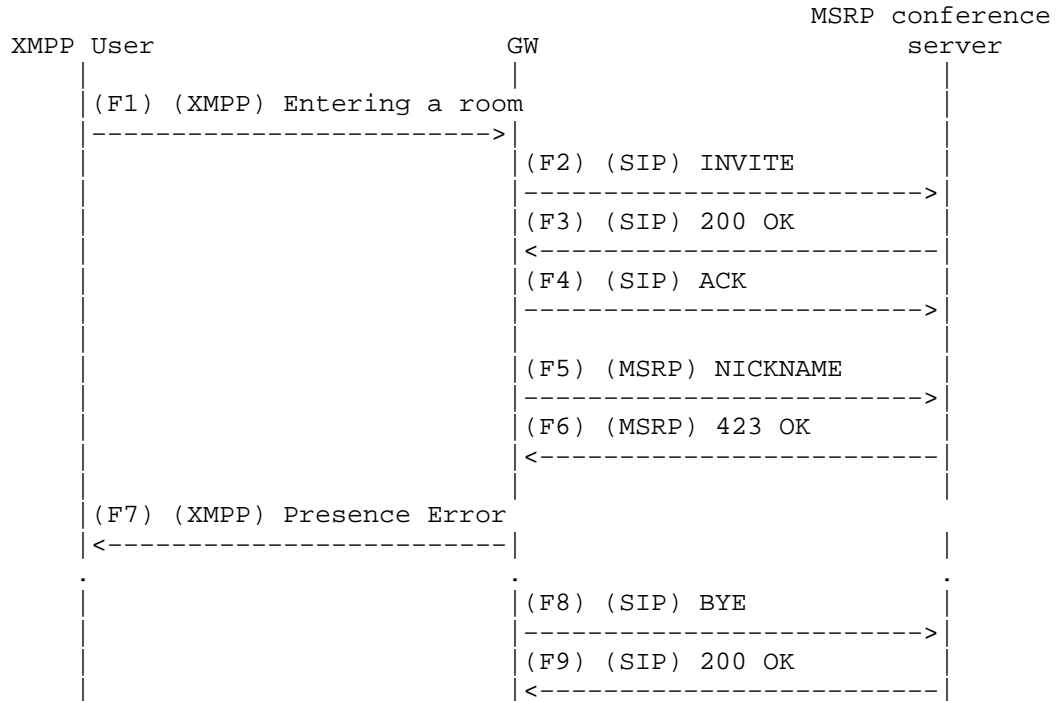
Juliet MAY include a custom exit message in the presence stanza of type "unavailable"

Example: (F16) Juliet exiting a chatroom

```
<presence from='juliet@example.com'
          to='verona@chat.shakespeare.net/juliet'
          type='unavailable'>
  <status>I can not chat now!</status>
</presence>
```

Upon receiving such stanza exiting the multi-user chat room, the XMPP-to-SIP gateway MUST before delivering the message and then, after the message is successfully delivered, it terminates the SIP session by sending a SIP BYE to MSRP conference server. The MSRP conference server then responds with a 200 OK.

2.6. Nickname Conflict



The chat room server analyzes the existing allocation of nicknames, and detects that the nickname proposal is already provided to another participant by the conference. In this case the MSRP conference server answers with a 423 response.

Example: (F6) the chat room does not accept the nickname proposal

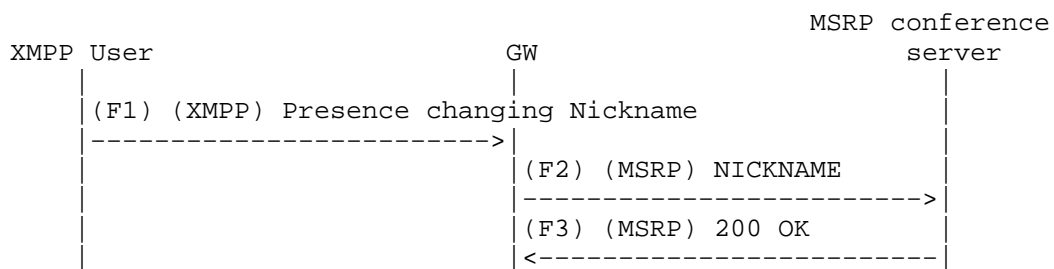
```
MSRP a786hjs2 423 Nickname usage failed
To-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
From-Path: msrp://s2x.shakespeare.net:12763/kjhd37s2s20w2a;tcp
-----a786hjs2
```

Upon receiving such a response, the SIP-to-XMPP gateway MUST translate it in an xmpp presence stanza of type "error" specifying a <conflict/> error condition.

Example: (F7) Juliet sends a Message to all occupants

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

2.7. Changing Nickname



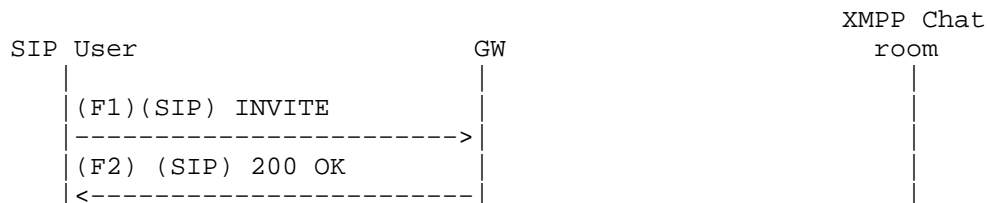
If Juliet decides to changing her nickname within the room, she SHOULD send an update presence information to the room, specifically she SHOULD send a new Nickname in the same room.

Example: (F1) Juliet changing the nickname

```
<presence from='juliet@example.com'
          to='verona@chat.shakespeare.net/July'>
</presence>
```

3. MSRP Multiparty Instant Message (IM) Session to XMPP Group Chat

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





3.1. Entering a Room

When the MSRP user ("Romeo") wants to join a multi-user chat 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 specifying support for a Message/CPIM top level wrapper for the MSRP message.

Example: (F1) SIP user starts the session

```
INVITE sip:verona@chat.shakespeare.net SIP/2.0
To: <sip:verona@chat.shakespeare.net>
From: <sip:romeo@example.com>;tag=786
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/ansp7lweztas;tcp
a=chatroom:nickname private-message
```

[OPEN ISSUE: 1] [I-D.ietf-simple-chat] does not say anything about the inclusion of the SDP "chatroom" attribute in the INVITE however that is the only way for a GW to understand the the INVITE is establishing a group-chat session

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 successfully with a SIP 200 OK (F2), but it MUST NOT yet translate the request into an XMPP presec stanza before the MSRP user set up the nickname.

```
SIP/2.0 200 OK
To: <sip:verona@chat.shakespeare.net>
From: <sip:romeo@example.com>;tag=786
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
```

[OPEN ISSUE: 1] the GW could use a temporary nick name and translate directly the request into a XMPP presence stanza, entering the XMPP chat room

Example: (F4) the MSRP user set up the nickname

```
MSRP a786hjs2 NICKNAME
To-Path: path:msrp://s2x.example.net:7313/ansp71weztas;tcp
From-Path: 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 to generate an XMPP presence stanza and sending it to the hostname hosting that chat room.

Example: (F5) Romeo entering a chatroom

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

If the room does not already contain another user with the nickname, the service accept the access. So if the GW 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)

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

3.2. Presence Broadcast

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

Example: (F7) the chat room presence information translated in XMPP

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

Upon receiving such a response, if the MSRP has already completed the subscription to the Conference Event package [RFC4575], the XMPP-to-SIP gateway MUST translate it in a SIP NOTIFY request.

Example: (F10) the XMPP-to-SIP notifies the presence information

```
NOTIFY sip:romeo@example.com SIP/2.0
To: Juliet <sip:romeo@example.com>;tag=43524545
From: <sip:verona@chat.shakespeare.net>;tag=a3343df32
Call-ID: k3l43id034ksererff
Event: conference
Subscription-State: active;expires=3600
Content-Type: application/conference-info+xml
Content-Length: ...
```

```
<conference-info version="0" state="full"
  entity="sip:3402934234@conf.example.com">
  <conference-description>
    <subject>Today in Verona</subject>
    <conf-uris>
      <entry>
        <uri>tel:+18882934234</uri>
      </entry>
    </conf-uris>
  </conference-description>
  <users>
    <user entity="sip:juliet@example.com" state="full">
      <nickname-text>juliet</nickname-text>
      <roles>
        <entry>participant</entry>
      </roles>
    </user>
  </users>
</conference-info>
```

3.3. Exchanging 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/>

3.3.1. Sending a Message to All Occupants

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

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

```
MSRP a786hjs2 SEND
To-Path: path:msrp://s2x.example.net:7313/ansp7lweztas;tcp
From-Path: path:msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
Message-ID: 87652492
Byte-Range: 1-*/*
Content-Type: message/cpim

To: <sip:verona@chat.shakespeare.net;transport=tcp>
From: <sip:juliet@example.com>
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 in a xmpp message stanza (F13) and then generate and send an MSRP response (F14).

The following examples show an exchange of a public message.

Example: (F13) Romeo sends a Message to all occupants

```
<message from='romeo@example.com'
  to='verona@chat.shakespeare.net'
  type='groupchat'>
  <body>Romeo is here!</body>
</message>
```

Example: (F14) the SIP-to-XMPP send the MSRP response

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

[OPEN ISSUE: 1] The SIP-to-XMPP gateway will receive back the echo message from the Chat room service. The SIP-to-XMPP gateway has to translate it back to the MSRP user or no?

3.3.2. Sending a Private Message

Romeo MAY send a "private message" to a selected occupant via the chat room service by sending a message to the occupant's room nick name.

The following examples show an exchange of a private message.

Example: (F12) Romeo sends a private message

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

To: <sip:juliet@chat.shakespeare.net>
From: <sip:romeo@example.com>
DateTime: 2008-10-15T15:02:31-03:00
Content-Type: text/plain

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

Example: (F13) Juliet sends a private message

```
<message from='romeo@example.com'
  to='verona@chat.shakespeare.net/juliet'
  type='chat' />
  <body>I am here!!!</body>
</message>
```

3.4. Exiting a Room

If Romeo decides to exit the multi-user chat room, his client sends SIP BYE to the <verona@chat.shakespeare.net> chat room.

Example: (F11) Romeo terminates the session

```

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

```

Upon receiving the SIP BYE, the SIP-to-XMPP gateway translate it in a presence stanza (F19) and send it to the XMPP chat room service. Then the SIP-to-XMPP gateway responds with a 200 OK to the MSRP user.

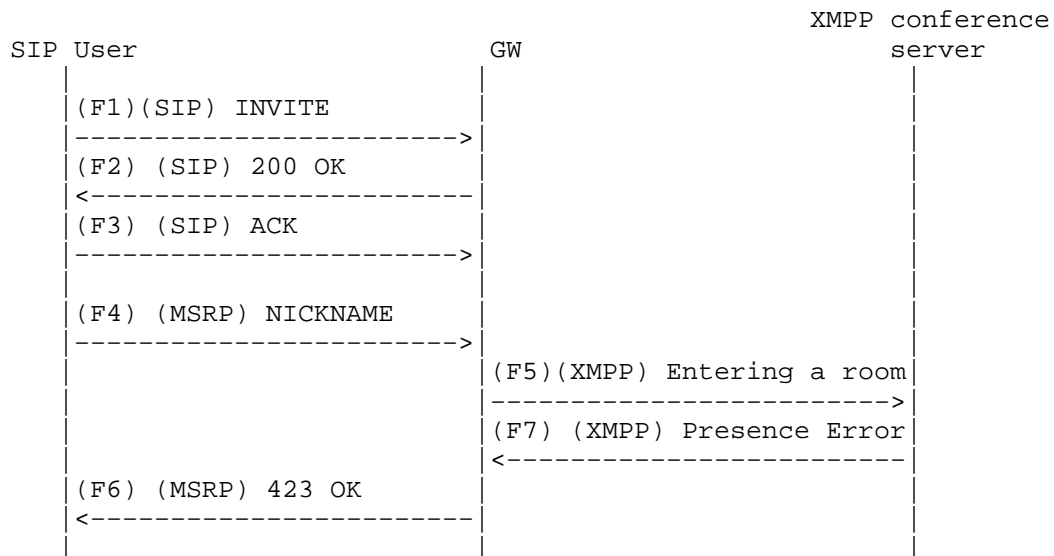
Example: (F19) Juliet exiting a chatroom

```

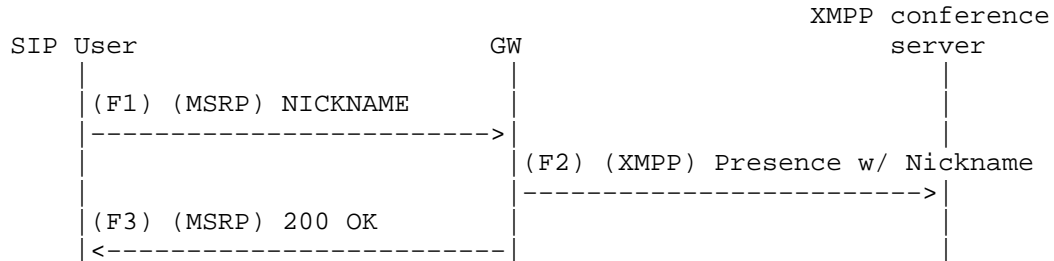
<presence from='romeo@example.com'
          to='verona@chat.shakespeare.net/romeo'
          type='unavailable'>
</presence>

```

3.5. Nickname Conflict



3.6. Changing Nickname



If Romeo decides to changing her nickname within the room, he SHOULD 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.

Example: (F1) the MSRP user changes the nickname

```

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

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

Example: (F2) Juliet changing the nickname

```

<presence from='juliet@example.com'
          to='verona@chat.shakespeare.net/montecchi'>
</presence>
  
```

4. Security Considerations

To follow.

5. IANA Considerations

This document requests no actions of IANA.

6. References

6.1. 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.
- [RFC3861] Peterson, J., "Address Resolution for Instant Messaging and Presence", RFC 3861, 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.
- [XEP-0045] Saint-Andre, P., "Multi-User Chat", XSF XEP 0045, July 2008.

6.2. Informative References

- [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-16 (work in progress), August 2012.
- [I-D.saintandre-sip-xmpp-chat] Saint-Andre, P., Gavita, E., Hossain, N., and S. Loreto, "Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): One-to-One Text Chat", draft-saintandre-sip-xmpp-chat-04 (work in progress), October 2012.
- [I-D.saintandre-sip-xmpp-core] Saint-Andre, P., Hourii, A., and J. Hildebrand, "Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Core", draft-saintandre-sip-xmpp-core-02 (work in progress), October 2012.

progress), October 2012.

[I-D.saintandre-sip-xmpp-im]

Saint-Andre, P., Hourri, A., and J. Hildebrand,
"Interworking between the Session Initiation Protocol
(SIP) and the Extensible Messaging and Presence Protocol
(XMPP): Instant Messaging",
draft-saintandre-sip-xmpp-im-02 (work in progress),
October 2012.

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

[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.

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

Fabio Forno
Bluendo srl
Via Morosini 10
Torino 10128
Italy

Email: fabio@bluendo.com

