

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.

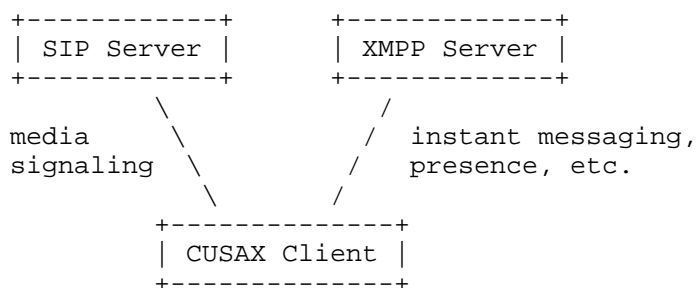


Figure 1: Division of Responsibilities

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-groupchat-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-groupchat-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

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

P. Saint-Andre
Cisco Systems, Inc.
E. Gavita
N. Hossain
S. Loreto
Ericsson
October 15, 2012

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

Abstract

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

Status of this Memo

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	6
1.6. Connection Maintenance	6
1.7. Acknowledgements	7
1.8. Discussion Venue	7
2. XMPP Formal Chat Session to MSRP	7
2.1. Initiating a Formal Session	8
2.2. Accepting a Formal Session	11
2.3. Exchanging Messages	13
2.4. Terminating a Formal Session	16
2.5. Cancelling the Negotiation	17
2.6. Rejecting a Formal Session	19
3. XMPP Informal Session to MSRP	20
4. MSRP to XMPP Formal Session	24
4.1. Initiating a Session	25
4.2. Accepting a Session	28
4.3. Completing the Transaction	29
4.4. Exchanging Messages	29
4.5. Terminating a Session	31
4.6. Cancelling the Transaction	33
4.7. Rejecting the Transaction	34
4.8. Session Negotiation Fails	34
5. MSRP to XMPP Informal Session	35
6. Security Considerations	38
7. IANA Considerations	38
8. References	38
8.1. Normative References	38
8.2. Informative References	39
Authors' Addresses	40

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 one-to-one 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]. This document specifies mappings for one-to-one text chat sessions (sometimes called "session-mode" messaging); in particular, this document specifies mappings between XMPP and the Message Session Relay Protocol [RFC4975]. Mapping of multi-user text chat (sometimes called "groupchat") is out of scope for this document.

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 end users to send "instant messages" to other end users. The term "instant message" usually refers to messages sent between two end users for delivery in close to real time (rather than messages that are stored and forwarded to the intended recipient upon request). Generally, there are three kinds of instant messages:

1. Single messages, which are sent from the sender to the recipient outside the context of any one-to-one chat session or multi-user text conference. The message is immediately delivered and not stored in an inbox. In XMPP a single message is a <message/> stanza of type "normal" as specified in [RFC6121]. In SIP/SIMPLE a single message is sent via the MESSAGE method as specified in [RFC3428].
2. One-to-one chat messages, which are sent from the sender to the recipient (i.e., one-to-one) in the context of a "chat session" between the two entities. In XMPP a chat message is a <message/> stanza of type "chat". In SIP/SIMPLE a chat message is sent

- using an MSRP session as specified in [RFC4975].
3. Groupchat 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 [MSRP-MULTI].

This document covers only scenario #2 for converting XMPP messages of type "chat" to and from their corresponding SIP INVITE and MSRP message types on the SIP/SIMPLE side.

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 "chat" are specified in [RFC6121].
- o A method for formally negotiating an XMPP chat session is specified in the Stanza Session Negotiation extension to XMPP [XEP-0155].
- o SIP-based chat sessions using the SIP INVITE and SEND request types are specified in [RFC4975].

1.4. Formal and Informal Sessions

The traditional model for a one-to-one chat "session" in XMPP is for a user to simply send a message of type "chat" to a contact, without any formal negotiation of session parameters; the contact would then reply to the message, and the sum total of such messages exchanged during a defined period of time can be considered a chat session. This informal approach to chat sessions in XMPP can be mapped both to SIP pager-mode messaging using the SIP MESSAGE method (as documented in [I-D.saintandre-sip-xmpp-core]) and to an MSRP chat session. How a gateway chooses to map the XMPP chat session to the SIP side is a matter of the implementation, although guidelines are provided under Section 1.5.

However, in XMPP it is also possible to formally request a chat session and negotiate its parameters (e.g., security, privacy, message logging) before beginning the session and exchanging messages. The protocol for doing so is defined in [XEP-0155]. In this case, the XMPP chat session SHOULD be translated into an MSRP session.

This document covers the mapping of both informal and formally-negotiated XMPP chat sessions into MSRP sessions, and from MSRP sessions into XMPP informal and formal sessions.

1.5. Gateway Heuristics

When a gateway receives a chat message or chat session request intended for a recipient that is registered with the gateway itself or has an account on a local service, it SHOULD adhere to the following process in determining whether to (1) initiate a formal chat session with the recipient, (2) initiate an informal chat session with the recipient, or (3) return an error to the sender.

1. If the gateway has knowledge of the recipient's online endpoints (available resources in XMPP or registered UAs in SIP), then it SHOULD discover the capabilities of those endpoints.
2. If the gateway determines that one or more of the endpoints supports formal chat sessions, it SHOULD initiate a formal chat session with one of those endpoints (deciding among the endpoints based on presence information or communications priority).
3. If the gateway determines that none of the endpoints supports formal chat sessions, it SHOULD initiate an informal chat session with one of those endpoints (deciding among the endpoints based on presence information or communications priority).
4. If the gateway does not know if the recipient has any online endpoints, it SHOULD return an appropriate error to the sender.

The methods by which a gateway determines support for various capabilities are protocol-specific. For XMPP a gateway SHOULD use the Service Discovery extension defined in [XEP-0030] or, if it receives presence information from the XMPP endpoint, use the Entity Capabilities extension defined in [XEP-0115]. For MSRP a gateway SHOULD use the Session Description Protocol defined in [RFC4566] in conjunction with a high-level protocol that provides a capability query, such as the SIP OPTIONS request defined in [RFC3261].

1.6. Connection Maintenance

XMPP makes use of long-lived TCP connections. If mobility affecting Layer 3 causes a dropped connection, the connection must be re-established. If mobility does not preserve the IP address, the TCP connection will be dropped. Any TLS session and SASL associations must be re-established if the TCP connection is dropped. XMPP binds directly to TCP in the core specification, so the TCP session must remain open for the entire duration of the chat session. The XMPP Standards Foundation does define protocol extensions enabling transport of XMPP traffic over HTTP (refer to [XEP-0124] and [XEP-0206]), so that individual messages are carried using HTTP and

are more robust in environments such as mobile networks, allowing for better recovery if a TCP session is broken.

SIMPLE is similar when using MSRP. The Message Session Relay Protocol [RFC4975] is a protocol for transmitting instant messages (IM) in the context of a session. The protocol specification describes how the session can be negotiated and established with an offer or answer (see [RFC3264]) using the Session Description Protocol [RFC4566]. In SIMPLE, this exchange is carried using SIP as the signaling protocol. After the TCP connection is established, if it fails for any reason, then an MSRP endpoint MAY choose to re-create such a session using a new SDP exchange in a SIP re-INVITE. SIMPLE also uses the MESSAGE request type for transporting instant messaging outside the context of a session. The MESSAGE request is sent inside the signaling path without establishing any dedicated connection.

1.7. Acknowledgements

Some text in this document was borrowed from [I-D.saintandre-sip-xmpp-core] and from [XEP-0155].

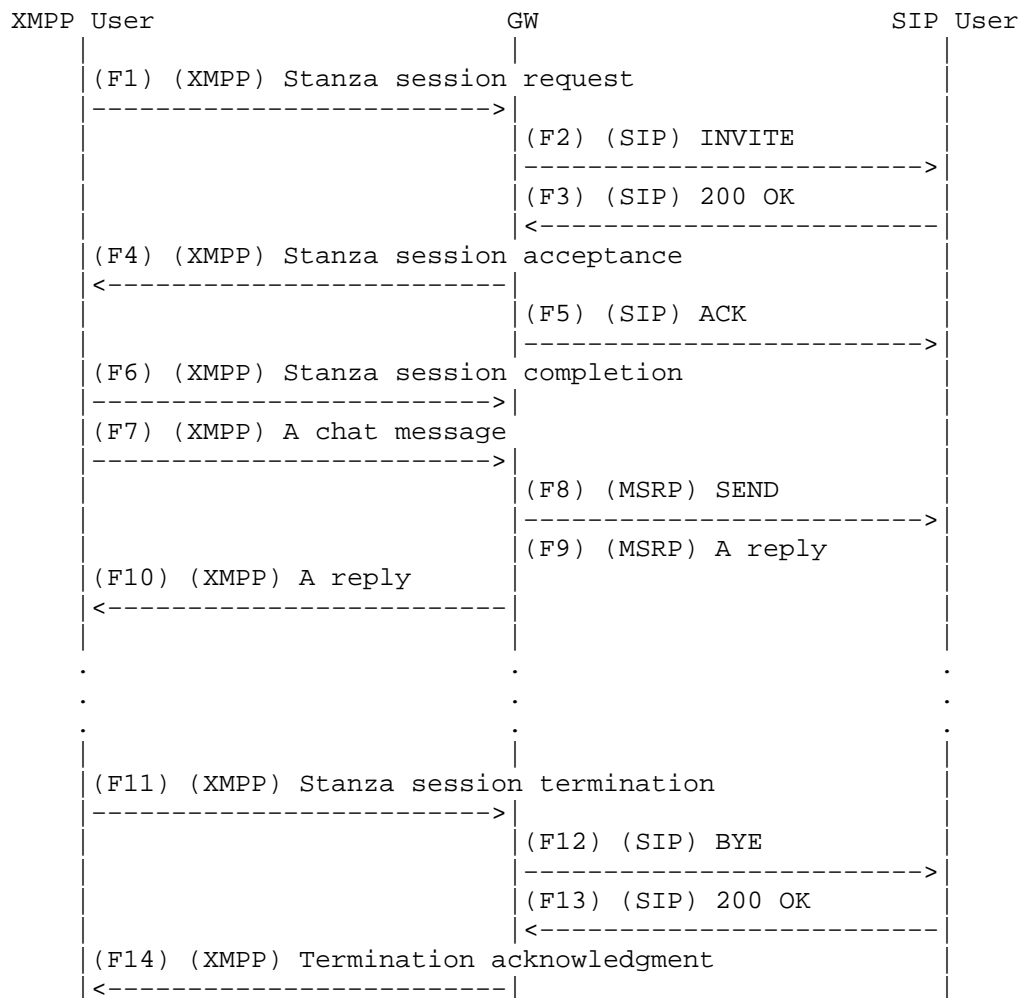
1.8. Discussion Venue

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 Formal Chat Session to MSRP

This section describes how to map an XMPP "formal session" to an MSRP session.

The XMPP formal session is based on the protocol described in [XEP-0155], which enables the initiation, renegotiation, and termination of a formal chat session on the XMPP side. This approach maps to the semantic of the SIP INVITE and BYE methods.



2.1. Initiating a Formal Session

When the XMPP user ("Juliet") wants to initiate a negotiated session with a SIP user ("Romeo"), she sends a <message/> stanza to Romeo containing a <feature/> child qualified by the 'http://jabber.org/protocol/feature-neg' namespace. The <message/> stanza must not contain a <body/> child (as specified in [RFC6121]), since that child element is used for human-readable text. The <message/> stanza type should be "normal". The stanza MUST contain a <thread/> element for tracking purposes (where the newly-generated ThreadID is unique to the proposed session). The encapsulated data form MUST contain a FORM_TYPE field whose type is "hidden" and whose value is "urn:xmpp:ssn"; it must also contain a boolean field named

"accept".

The XMPP user may request a session with a specific resource of the contact. However in this document the resource identifier will be ignored and discarded for cross-system interworking.

Example: (F1) Juliet starts a formal session

```
<message from='juliet@example.com'
        to='romeo@example.net'
        type='normal'>
  <thread>711609sa</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <title>Open chat with Juliet?</title>
      <field var='FORM_TYPE' type='hidden'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field label='Accept this session?'
            type='boolean'
            var='accept'>
        <value>true</value>
        <required/>
      </field>
      <field label='Primary written language of the chat'
            type='list-single'
            var='language'>
        <value>en</value>
        <option label='English'><value>en</value></option>
        <option label='Italiano'><value>it</value></option>
      </field>
      <field label='XHTML formatting'
            type='list-single'
            var='http://jabber.org/protocol/xhtml-im'>
        <value>may</value>
        <option label='Allow XHTML formatting'>
          <value>may</value>
        </option>
        <option label='Disallow XHTML formatting'>
          <value>mustnot</value>
        </option>
      </field>
    </x>
  </feature>
</message>
```

Upon receiving such a session request, the XMPP server to which Juliet has authenticated attempts to deliver the request to a local

user or attempts to route the request 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 IM-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 [RFC2782] lookups. For message stanzas, the order of lookups recommended by [RFC6121] is to first try the "_xmpp-server" service as specified in [RFC6120] and to then try the "_im" service as specified in [RFC3861]. Here we assume that the first lookup will fail but that the second lookup will succeed and return a resolution "_im._simple.example.net.", since we have already assumed that the example.net hostname is running a SIP instant messaging service. (Note: The XMPP server may have previously determined that the 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 message off to its local XMPP-to-SIP gateway (x2s.example.com), which transforms the message into SIP syntax and routes it to the remote SIMPLE server (example.net).

Example: (F2) Juliet starts a formal session (SIP transformation)

```
INVITE sip:romeo@example.net SIP/2.0
To: <sip:romeo@example.net>
From: <sip:juliet@example.com>;tag=786
Subject: Open chat with Juliet?
Call-ID: 711609sa
Content-Type: application/sdp
```

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

Here 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.

Native XMPP messages as described in [RFC6121] supports text (i.e., UTF-8) only. However, there exists an XMPP extension for XHTML-formatted messages, as defined by the XHTML-IM integration set specified in [XEP-0071]. Unless the use of XHTML-formatted messages is supported by the endpoints or negotiated during session establishment, the "accept-types" attribute that follows an MSRP media line SHOULD indicate "text/plain" as the only media-type that is acceptable to the endpoint; if XHTML is supported or negotiated, the "accept-types" attribute MAY also indicate a media-type of "text/html". (Note: The XHTML-IM integration set supports only a subset of XHTML formatting; it is the responsibility of a gateway to map between full XHTML and XHTML-IM.)

As specified in [I-D.saintandre-sip-xmpp-core], the mapping of XMPP syntax elements to SIP and SDP 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
<thread/>	Call-ID
from	From
to	To
<title/>	Subject
xml:lang	a=lang:<language tag>
-	a=accept-types:text/plain

2.2. Accepting a Formal Session

Here we assume that the SIP user agent that receives the SIP invitation (containing an offered session description that includes a session of MSRP) accepts the invitation and includes an answer session description that acknowledges the choice of media.

Example: (F3) Romeo accepts the request

```
SIP/2.0 200 OK
To: <sip:romeo@example.net>;tag=087js
From: <sip:juliet@example.com>;tag=786
Call-ID: 711609sa
Content-Type: application/sdp
```

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

Upon receiving such a response, the SIMPLE server or associated SIP-to-XMPP gateway SHOULD remember that this is a response to a SIP transaction related to an XMPP-SIP translation, based on the SIP Call-ID (which is functionally equivalent to the XMPP <thread/>). The SIP-to-XMPP gateway is responsible for translating the response into an XMPP message stanza and routing it from the SIP user to the XMPP server or associated XMPP-to-SIP gateway.

The SIP-to-XMPP gateway MUST include in the response translation values for all the fields that the XMPP request indicated are required.

Example: (F4) Romeo accepts the request (XMPP translation)

```
<message from='romeo@example.net'
  to='juliet@example.com'
  type='normal'>
  <thread>711609sa</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'><value>true</value></field>
      <field var='language'><value>it</value></field>
    </x>
  </feature>
</message>
```

The SIP-to-XMPP gateway MUST also send a SIP ACK to the SIP user.

Example: (F5) Gateway sends ACK to Romeo's UA

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

If Romeo accepted the session, Juliet MUST either complete or cancel the stanza session negotiation. The user's client SHOULD verify that the selected values of the fields are acceptable before completing the stanza session negotiation -- and confirming that the session is open -- by replying with the form 'type' attribute set to 'result'. The form MUST contain the FORM_TYPE field and the "accept" field set to "1" or "true".

Example: (F6) Juliet completes negotiation

```
<message from='juliet@example.com'
  to='romeo@example.net'
  type='normal'>
  <thread>711609sa</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'><value>true</value></field>
    </x>
  </feature>
</message>
```

Upon receiving such a stanza completing the session negotiation, the XMPP server MUST NOT send any confirmation to the SIP side; instead, it MUST route the acceptance to the SIMPLE server or associated SIP-to-XMPP gateway.

The session is now open and the parties can proceed to exchanging messages.

2.3. Exchanging Messages

Once the session is created, the endpoints can exchange an unbounded number of messages.

The XMPP 'id' attribute is not required in the protocol and there is no way to enforce its use for messages. It is RECOMMENDED to include it as a negotiable item in the SSN negotiation, via the "message-ids" field. However, it is possible that the 'id' will not be present

within the <message/> stanza; in this case the XMPP-to-SIP gateway MUST generate a new unique Message-ID.

If the XMPP user has not explicitly requested message receipts during the negotiation, it is RECOMMENDED that the SIP-to-XMPP gateway shall insert a Failure-Report header field value of "no" during the creation of a SEND request. The XMPP user can include a request for message receipts using the Message Receipts XMPP protocol extension [XEP-0184]; use of this extension can be negotiated via the "urn:xmpp:receipts" field during SSN negotiation.

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 2: Message syntax mapping from XMPP Message to MSRP

XMPP Element or Attribute	MSRP Header
to	To-Path
from	From-Path
<body/>	body of the SEND request
-	Content-Type: text/plain
id	Message-ID

The following examples show an exchange of messages.

Example: (F7) Juliet sends an XMPP message

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

Example: (F8) Gateway transforms XMPP message to MSRP

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

```
Art thou not Romeo, and a Montague?
-----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, Romeo's client MUST immediately generate and send a response.

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

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

Example: (F9) Romeo sends a reply

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

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

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

Example: (F10) Gateway transforms MSRP message to XMPP

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

Note: The size of the XML character data of an XMPP message is not limited by the protocol, but is sometimes limited in deployment. However messages sent using MSRP can be delivered in several SEND requests, so when the XMPP-to-SIP gateway receives a message longer than 2048, it is **STRONGLY RECOMMENDED** it delivers this message using as few chunks (at least 2048 octets long) as possible.

2.4. Terminating a Formal Session

If Juliet decides to terminate the negotiated chat session, her client sends a <message/> stanza to Romeo containing a data form of type "submit". The <message/> stanza MUST contain a <thread/> element with the same XML character data as the original initiation request. The data form containing a boolean field named "terminate" set to a value of "1" or "true".

Example: (F11) Juliet terminates the chat session

```
<message from='juliet@example.com'
        to='romeo@example.net'
        type='normal'>
  <thread>711609sa</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='terminate'>
        <value>1</value>
      </field>
    </x>
  </feature>
</message>
```

Upon receiving such stanza terminating the chat session, the XMPP-to-SIP gateway terminates the SIP session by sending a SIP BYE to tear down the MSRP session with Romeo's client. Romeo's SIP client then responds with a 200 OK.

Example: (F12) Juliet terminates the chat session (SIP translation)

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

Example: (F13) Romeo acknowledges termination

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

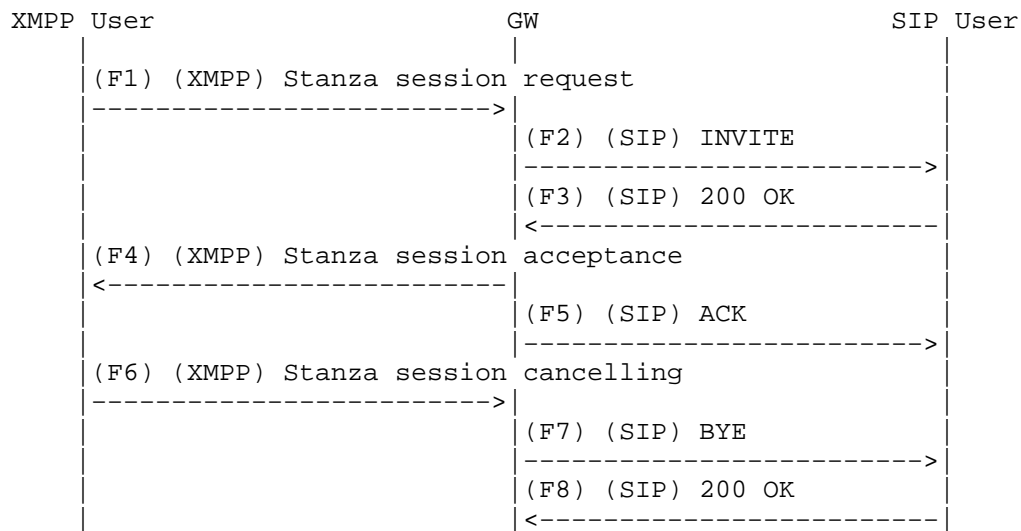
Upon receiving the 200 OK, the SIP-to-XMPP gateway acknowledges the termination of the chat session on the XMPP side by sending a <message/> containing a data form of type "result", and the value of the "terminate" field set to "1" or "true". The client must mirror the <thread/> value it received.

Example: (F14) Romeo acknowledges termination (XMPP translation)

```
<message from='romeo@example.net'
  to='juliet@example.com'
  type='normal'>
  <thread>711609sa</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='terminate'>
        <value>1</value>
      </field>
    </x>
  </feature>
</message>
```

2.5. Cancelling the Negotiation

If Romeo accepted the session but Juliet decides to cancel the stanza session negotiation, the flow is as follows.



That is, Juliet's client shall reply with a data form containing the FORM_TYPE field and the "accept" field set to "0" or "false":

Example: (F6) User cancels stanza session negotiation

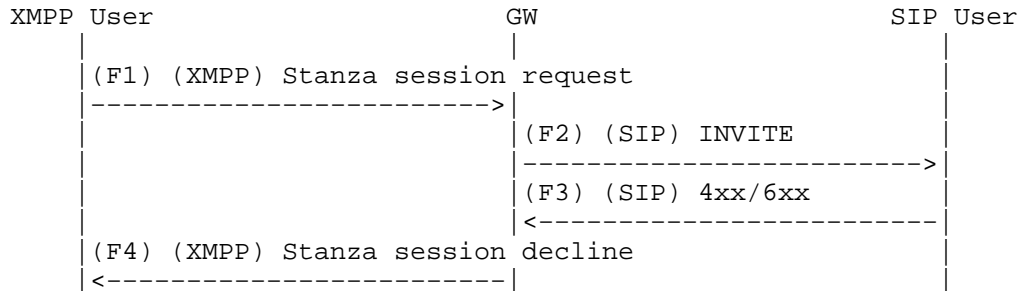
```
<message from='juliet@example.com'
  to='romeo@example.net'
  type='normal'>
  <thread>711609sa</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'>
        <value>0</value>
      </field>
    </x>
  </feature>
</message>
```

Upon receiving such stanza cancelling the session negotiation, the XMPP-to-SIP gateway MUST send a SIP BYE. Once the XMPP-to-SIP gateway receives the 200 OK, the internal session data is removed and the session is officially cancelled also in the SIP-to-XMPP gateway.

If the SIP user had sent any messages to XMPP while awaiting confirmation of the session, the SIP-to-XMPP gateway MUST return them to the SIP user with an appropriate error.

2.6. Rejecting a Formal Session

A common scenario occurs when the SIP UA is currently unwilling or unable to accept a formal session, in which case the flow is as follows.



Here the SIP UA declining an offer contained in an INVITE SHOULD return a 4xx or a 6xx response, such as 406 Not Acceptable or 603 Decline. Such a response SHOULD include a Warning header field value explaining why the offer was rejected.

Example: (F3) SIP user declines offer and specifies reason (SIP)

```

SIP/2.0 603 Decline
From: <sip:juliet@example.com>;tag=786
To: <sip:romeo@example.net>;tag=087js
Call-ID: 711609sa
Warning: I'm busy!
Content-Length: 0
  
```

Upon receiving the error response for the SIP INVITE, the XMPP-to-SIP gateway shall send a "Session Reject" message back to the XMPP Client. This message contains a data form that MUST contain the FORM_TYPE field and the "accept" field set to "0" or "false". It is RECOMMENDED that the form does not contain any other field even if the request indicated they are required. The client MAY include a reason in the <body/> child of the <message/> stanza. The content of the Warning header field present in the SIP response SHOULD be mapped to a "reason" field in the data form. If the Warning header is not present then the descriptive phrase of the SIP response can be used.

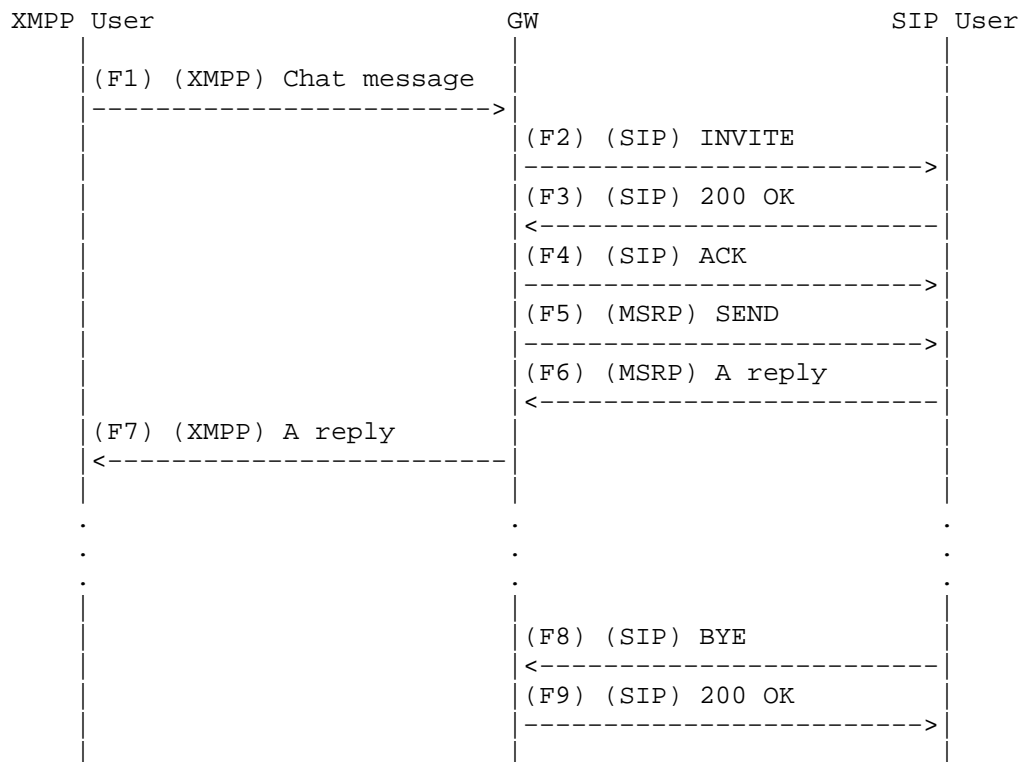
Example: (F4) SIP user declines offer and specifies reason (XMPP translation)

```
<message from='romeo@example.net'
  to='juliet@example.com'
  type='normal'>
  <thread>711609sa</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'>
        <value>0</value>
      </field>
      <field var='reason'>
        <value>I&apos;m busy!</value>
      </field>
    </x>
  </feature>
</message>
```

3. XMPP Informal Session to MSRP

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

The order of events is as follows.



First the XMPP user would generate an XMPP chat message.

Example: (F1) Juliet sends an XMPP message

```

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

```

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

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

```
INVITE sip:romeo@example.net SIP/2.0
To: <sip:romeo@example.net>
From: <sip:juliet@example.com>;tag=786
Subject: Open chat with Juliet?
Call-ID: 711609sa
Content-Type: application/sdp
```

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

Here we assume that Romeo accepts the MSRP session request.

Example: (F3) Romeo accepts the request

```
SIP/2.0 200 OK
To: <sip:romeo@example.net>;tag=087js
From: <sip:juliet@example.com>;tag=786
Call-ID: 711609sa
Content-Type: application/sdp
```

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

The XMPP-to-SIP gateway then acks the session acceptance on behalf of Juliet.

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

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

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

Example: (F5) Gateway transforms XMPP message to MSRP

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

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

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

Example: (F6) Romeo sends a reply

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

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

Note: As previously described, if the users have not negotiated the use message receipts, it is RECOMMENDED that the SIP-to-XMPP gateway shall insert a Failure-Report header field value of "no" during the creation of a SEND request.

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

Example: (F7) Gateway transforms MSRP message to XMPP

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

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

Example: (F8) Romeo terminates the chat session

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

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

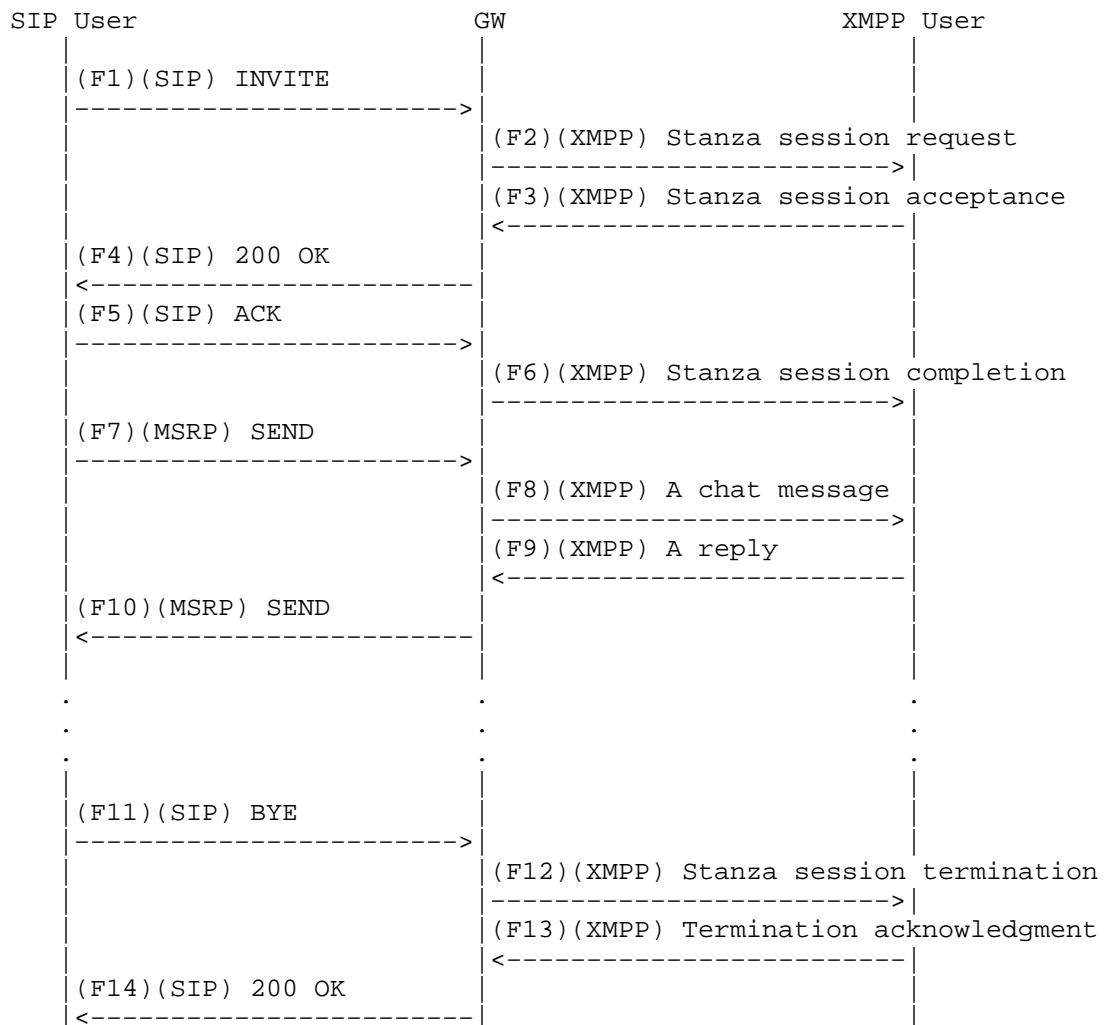
Example: (F9) Gateway acknowledges termination

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

4. MSRP to XMPP Formal Session

Unlike the XMPP protocol, the MSRP protocol offers only one way to initiate a chat session, typically using the Session Description Protocol [RFC4566] via the SIP offer/answer mechanism (see [RFC3264]).

The order of events is as follows.



4.1. Initiating a Session

When Romeo wants to start an MSRP message session with Juliet, 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" attribute and corresponding URIs. The MSRP media line is also accompanied by an "accept-types" attribute used to specify the only media-types acceptable for Romeo (i.e., text/plain and/or text/html).

Note: In addition to plain text messages, MSRP is able to carry arbitrary (binary) Multipurpose Internet Mail Extensions [RFC2045]

compliant content, such as images or video clips. Disposition of media types other than text/plain and text/html is out of scope for this specification and is a matter of implementation.

Example: (F1) SIP user starts the session

```
INVITE sip:juliet@example.com SIP/2.0
To: <sip:juliet@example.com>
From: <sip:romeo@example.net>;tag=576
Subject: Open chat with Romeo?
Call-ID: 742507no
Content-Type: application/sdp
```

```
c=IN IP4 s2x.example.net
m=message 7313 TCP/MSRP *
a=accept-types:text/plain
a=lang:en
a=lang:it
a=path:msrp://s2x.example.net:7313/ansp7lweztas;tcp
```

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 is responsible for translating the request into an XMPP message stanza to initiate a negotiated session from the SIP user to the XMPP user.

Example: (F2) SIP user starts the session (XMPP transformation)

```
<message from='romeo@example.net'
  to='juliet@example.com'
  type='normal'>
  <thread>742507no</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='form'>
      <title>Open chat with Romeo?</title>
      <field var='FORM_TYPE' type='hidden'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field label='Accept this session?' type='form' var='accept'>
        <value>true</value>
        <required/>
      </field>
      <field label='Primary written language of the chat'
        type='list-single'
        var='language'>
        <value>en</value>
        <option label='English'><value>en</value></option>
        <option label='Italiano'><value>it</value></option>
      </field>
    </x>
  </feature>
</message>
```

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 in the foregoing table are undefined.)

Table 3: Message syntax mapping from SIP to XMPP

SIP Header or SDP Contents	XMPP Element or Attribute
Call-ID	<thread/>
From	from
To	to
Subject	<title/>
accept-types	-
a=lang	xml:lang
To	to

See previous note regarding negotiation and use of the XHTML-IM integration set for XHTML-formatted messages (i.e., the "text/html" accept-type).

4.2. Accepting a Session

If the request is accepted then Juliet's client MUST include all the fields that were marked as required in the request message.

In the example below, we assume that Juliet accepts the session and specifies that she prefers to speak Italian with Romeo.

Example: (F3) Juliet accepts session and specifies parameters

```
<message from='juliet@example.com'
        to='romeo@example.net'
        type='normal'>
  <thread>742507no</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'><value>true</value></field>
      <field var='language'><value>it</value></field>
    </x>
  </feature>
</message>
```

Upon receiving such a response, the SIP-to-XMPP gateway SHOULD remember that this is a response to a stanza related to an SIP-XMPP translation, based on the SIP Call-ID. The SIP-to-XMPP gateway is responsible for translating the response into a SIP response and delivering it from the XMPP user back to the SIP user.

Example: (F4) Juliet accepts session (SIP translation)

```
SIP/2.0 200 OK
To: <sip:juliet@example.com>;tag=534
From: <sip:romeo@example.net>;tag=576
Call-ID: 742507no
Content-Type: application/sdp

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

4.3. Completing the Transaction

In this case, the 200 OK is routed back and is received by Romeo's UA. Finally, Romeo's client sends an acknowledgment message, ACK, to Juliet's client to confirm the reception of the final response (200 OK).

Example: (F5) Romeo sends ACK

```
ACK sip:juliet@example.com SIP/2.0
To: <sip:juliet@example.com>;tag=534
From: <sip:romeo@example.net>;tag=576
Call-ID: 742507no
```

Upon receiving the ACK, the SIP-to-XMPP gateway SHOULD remember this is an acknowledgment to an XMPP formal session. The SIP-to-XMPP gateway is responsible for translating the acknowledgment into a confirmation stanza, without inserting other content (e.g. a <body/> element cannot be inserted).

Example: (F6) Romeo sends ACK (XMPP translation)

```
<message from='romeo@example.net'
  to='juliet@example.com'
  type='normal'>
  <thread>742507no</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'>
        <value>true</value>
      </field>
    </x>
  </feature>
</message>
```

4.4. Exchanging Messages

When Romeo wants to send a message, he creates an MSRP SEND request that contains the message.

Example: (F7) Romeo sends a message

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

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

Upon receiving the MSRP SEND request, the SIP-to-XMPP gateway SHOULD remember that the message is for an XMPP user. The SIP-to-XMPP gateway is responsible for translating the MSRP SEND request into an XMPP message stanza.

Example: (F8) Romeo sends a message (XMPP translation)

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

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

MSRP Header	XMPP Element or Attribute
To-Path	to
From-Path	from
body of the SEND request	<body/>
Content-Type: text/plain	-
Message-ID	id

Upon receiving the chat message, Juliet can send a reply.

Example: (F9) Juliet sends a reply

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

Example: (F10) Gateway transforms XMPP message to MSRP

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

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

4.5. Terminating a Session

When Romeo wants to terminate the session, he is required to send a SIP BYE request.

Example: (F11) Romeo terminates the session

```
BYE juliet@example.com sip: SIP/2.0
Max-Forwards: 70
From: <sip:romeo@example.net>;tag=576
To: <sip:juliet@example.com>;tag=534
Call-ID: 742507no
Cseq: 1 BYE
Content-Length: 0
```

Upon receiving the SIP BYE request, the XMPP-to-SIP gateway SHOULD translate the request to a <message/> stanza containing a data form of type "submit". The <message/> element MUST contain a <thread/> element with the same XML character data as the original initiation request. The data form containing a boolean field named "terminate" should be set to a value of "1" or "true".

Example: (F12) Romeo terminates the session (XMPP translation)

```
<message from='romeo@example.net'
  to='juliet@example.com'
  type='normal'>
  <thread>742507no</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='terminate'>
        <value>1</value>
      </field>
    </x>
  </feature>
</message>
```

Juliet explicitly acknowledges the termination of the chat session on the XMPP side by sending a <message/> containing a data form of type "result", and the value of the "terminate" field set to "1" or "true". The client MUST mirror the <thread/> value it received.

Example: (F13) Juliet acknowledges the termination of the session

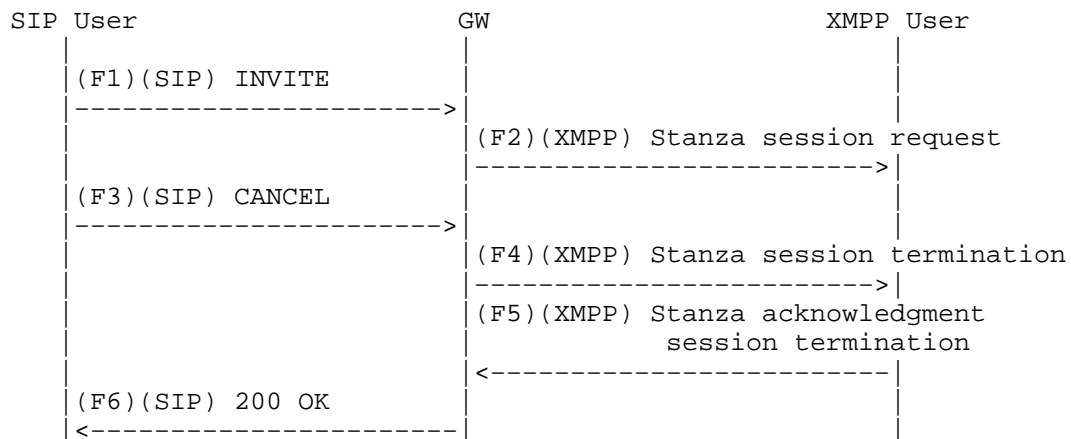
```
<message from='juliet@example.com'
  to='romeo@example.net'
  type='normal'>
  <thread>742507no</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='terminate'>
        <value>1</value>
      </field>
    </x>
  </feature>
</message>
```

Upon receiving the acknowledgment message, the XMPP-to-SIP gateway SHOULD translate it to a SIP answer 200 OK.

Example: (F14) Juliet acknowledges the termination of the session (SIP translation)

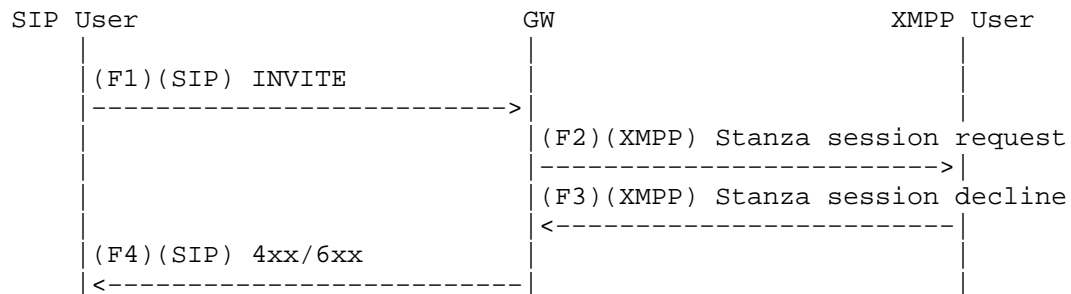
```
SIP/2.0 200 OK
From: <sip:romeo@example.net>;tag=576
To: <sip:juliet@example.com>;tag=534
Call-ID: 742507no
CSeq: 1 BYE
```

4.6. Cancelling the Transaction



A common scenario occurs when the SIP user issues an invitation to set up a chat session with an XMPP user and immediately after the SIP invitation is sent, the SIP user decides to cancel it. The SIP-to-XMPP gateway will receive the CANCEL request and using the Call-ID, To, From and CSeq (sequence number only) header field values as a guide, will issue an XMPP stanza session termination request to the XMPP user to cancel the XMPP formal session (assuming that it was already set up). Once the XMPP-to-SIP gateway receives an ACK stanza message for the session termination, the XMPP-to-SIP gateway will respond with a status of 200 (OK) back to the SIP user. It is important to note that if the SIP session transaction does not exist, the XMPP-to-SIP gateway will return a status of 481 (Transaction Does Not Exist) back to the SIP User.

4.7. Rejecting the Transaction



Another common scenario occurs when the XMPP UA is currently not willing or able to accept a formal session request. The XMPP UA SHOULD decline the invitation. The data form MUST contain the FORM_TYPE field and the "accept" field set to "0" or "false". It is RECOMMENDED that the form does not contain any other fields even if the request indicated they are required.

Example: (F3) User declines offer

```

<message from='juliet@example.com'
  to='romeo@example.net'
  type='normal'>
  <thread>742507no</thread>
  <feature xmlns='http://jabber.org/protocol/feature-neg'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'>
        <value>urn:xmpp:ssn</value>
      </field>
      <field var='accept'><value>0</value></field>
    </x>
  </feature>
</message>
  
```

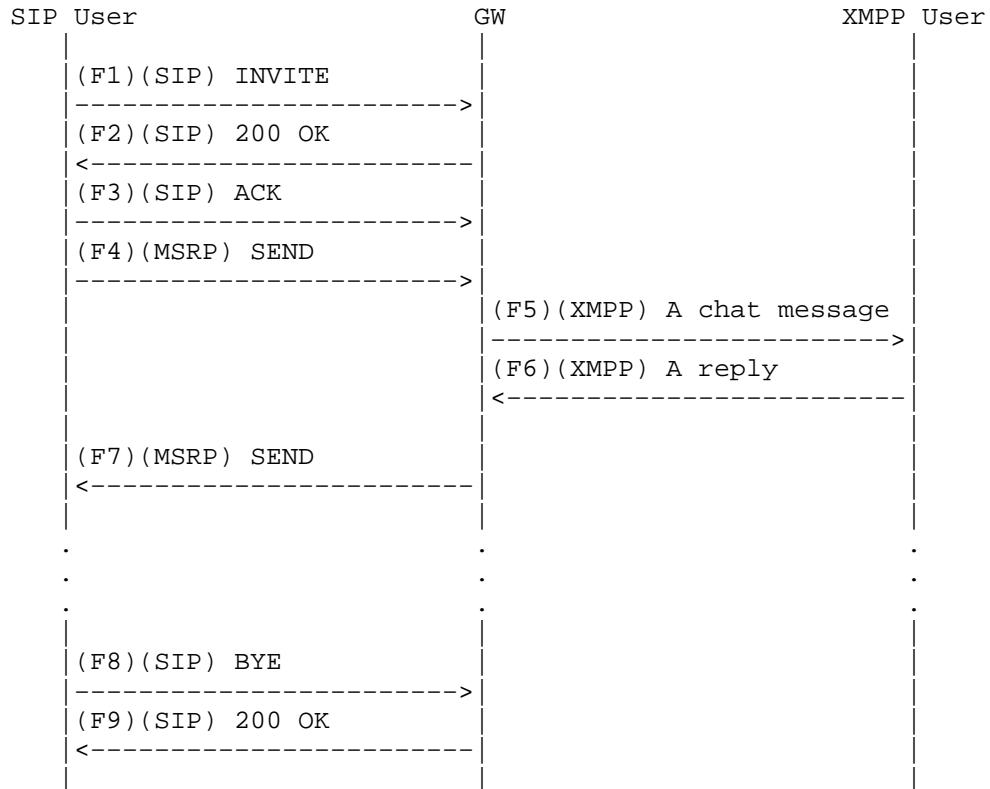
Upon receiving the declined response for the XMPP formal session request, the XMPP-to-SIP gateway SHOULD return a 4xx or a 6xx SIP response back to the SIP client.

4.8. Session Negotiation Fails

If the XMPP recipient of a formal session request does not support stanza session negotiation as specified in [XEP-0155], it will return an XMPP <service-unavailable/> stanza error. Upon receiving this error from the XMPP recipient, the XMPP-to-SIP gateway SHOULD return a 501 SIP response back to the SIP sender.

5. MSRP to XMPP Informal Session

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



Example: (F1) SIP user starts the session

```
INVITE sip:juliet@example.com SIP/2.0
To: <sip:juliet@example.com>
From: <sip:romeo@example.net>;tag=576
Subject: Open chat with Romeo?
Call-ID: 742507no
Content-Type: application/sdp
```

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

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

```
SIP/2.0 200 OK
To: <sip:juliet@example.com>;tag=534
From: <sip:romeo@example.net>;tag=576
Call-ID: 742507no
Content-Type: application/sdp
```

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

Example: (F3) Romeo sends ACK

```
ACK sip:juliet@example.com SIP/2.0
To: <sip:juliet@example.com>;tag=534
From: <sip:romeo@example.net>;tag=576
Call-ID: 742507no
```

Example: (F4) Romeo sends a message

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

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

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

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

Example: (F6) Juliet sends a reply

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

Example: (F8) Gateway transforms XMPP message to MSRP

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

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


Example: (F9) Romeo terminates the session

```
BYE juliet@example.com sip: SIP/2.0
Max-Forwards: 70
From: <sip:romeo@example.net>;tag=576
To: <sip:juliet@example.com>;tag=534
Call-ID: 742507no
Cseq: 1 BYE
Content-Length: 0
```

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

```
SIP/2.0 200 OK
From: <sip:romeo@example.net>;tag=576
To: <sip:juliet@example.com>;tag=534
Call-ID: 742507no
CSeq: 1 BYE
```

6. Security Considerations

To follow.

7. IANA Considerations

This document requests no actions of IANA.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, 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-0155] Paterson, I. and P. Saint-Andre, "Stanza Session Negotiation", XSF XEP 0155, January 2008.

8.2. Informative References

- [I-D.saintandre-sip-xmpp-core] Saint-Andre, P., Hourì, 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.
- [I-D.saintandre-sip-xmpp-im] Saint-Andre, P., Hourì, 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-01 (work in progress), March 2009.
- [MSRP-MULTI] 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.
- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
- [RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [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, April 2007.
- [XEP-0071] Saint-Andre, P., "XHTML-IM", XSF XEP 0071, August 2007.
- [XEP-0115] Hildebrand, J., Saint-Andre, P., Troncon, R., and J. Konieczny, "Entity Capabilities", XSF XEP 0115, February 2008.
- [XEP-0124] Paterson, I., Smith, D., and P. Saint-Andre, "Bidirectional-streams Over Synchronous HTTP (BOSH)", XSF XEP 0124, October 2008.
- [XEP-0184] Saint-Andre, P. and J. Hildebrand, "Message Receipts", XSF XEP 0184, September 2007.
- [XEP-0206] Paterson, I., "XMPP Over BOSH", XSF XEP 0206, October 2008.

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

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

Email: eddy.gavita@ericsson.com

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

Email: Nazin.Hossain@ericsson.com

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Salvatore.Loreto@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 9, 2013

P. Saint-Andre
Cisco Systems, Inc.
A. Hour
IBM
J. Hildebrand
Cisco Systems, Inc.
February 5, 2013

Interworking between the Session Initiation Protocol (SIP) and the
Extensible Messaging and Presence Protocol (XMPP): Addresses and Error
Conditions
draft-saintandre-sip-xmpp-core-03

Abstract

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

Status of this Memo

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 August 9, 2013.

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	3
2. Terminology	3
3. Discussion Venue	3
4. Architectural Assumptions	3
5. Address Mapping	4
5.1. Overview	4
5.2. Local Part Mapping	5
5.3. Instance-Specific Mapping	7
5.4. SIP to XMPP	7
5.5. XMPP to SIP	8
6. Error Condition Mapping	9
6.1. XMPP to SIP	9
6.2. SIP to XMPP	9
7. Security Considerations	11
8. IANA Considerations	11
9. References	11
9.1. Normative References	11
9.2. Informative References	12
Appendix A. Acknowledgements	12
Authors' Addresses	13

1. Introduction

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

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

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

2. Terminology

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

3. Discussion Venue

The discussion venue for this document is the mailing list of the DISPATCH WG; visit <https://www.ietf.org/mailman/listinfo/dispatch> for subscription and archive information.

4. Architectural Assumptions

Protocol translation between SIP and XMPP could occur in a number of different entities, depending on the architecture of real-time communication deployments. For example, protocol translation could occur within a multi-protocol server, within a multi-protocol client, or within a gateway that acts as a dedicated protocol translator.

This document assumes that the protocol translation will occur within a gateway. (This assumption not meant to discourage protocol translation within multi-protocol clients or servers; instead, this

assumption is followed mainly to clarify the discussion and examples so that the protocol translation principles can be more easily understood and can be applied by client and server implementors with appropriate modifications to the examples and terminology.) Specifically, we assume that the protocol translation will occur within an "XMPP-to-SIP gateway" that translates XMPP syntax and semantics on behalf of an XMPP service when communicating with SIP services and/or within a "SIP-to-XMPP gateway" that translates SIP syntax and semantics on behalf of a SIP service when communicating with XMPP services.

This document assumes that a gateway will translate directly from one protocol to the other. We further assume that protocol translation will occur within a gateway in the source domain, so that information generated by the user of an XMPP service will be translated by a gateway within the trust domain of that XMPP service, and information generated by the user of a SIP service will be translated by a gateway within the trust domain of that SIP service.

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

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

```

#####
#                                     #
#   +-----+-----+   #   +-----+-----+   #
#   | example.net | GW |---#---| GW | example.com |   #
#   +-----+-----+   #   +-----+-----+   #
#           |               #           |               #
#   roмео@example.net       #       juliet@example.com   #
#                           #                           #
#####

```

5. Address Mapping

5.1. Overview

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

identified by a 'pres:' URI as specified in the Common Profile for Presence [RFC3859] (see [RFC3856]).

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

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

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

5.2. Local Part Mapping

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

```

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

Here we make the simplifying assumption that the local part of an

im:/pres: URI inherits from the "dot-atom-text" rule in [RFC5322] rather than the more complicated "local-part" rule:

[illegible]

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

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

Table 1: Allowed and disallowed characters

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

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

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

5.3. Instance-Specific Mapping

The meaning of a resourcepart in XMPP (i.e., the portion of a JID after the slash character, such as "foo" in the JID "user@example.com/foo") matches that of a Globally Routable User Agent URI (GRUU) in SIP [RFC5627]. In both cases, these constructs identify a particular device associated with the bare JID ("user@host") of an XMPP entity or with the Address of Record (AOR) of a SIP entity. Therefore, it is reasonable to map the value of a "gr" URI parameter to an XMPP resource part, and vice-versa.

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

5.4. SIP to XMPP

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

1. Remove URI scheme.
2. Split at the first '@' character into local part and hostname (mapping the latter is out of scope).
3. Translate any percent-encoded strings ("%hexhex") to percent-decoded octets.
4. Treat result as a UTF-8 string.

5. Translate "&" to "%26", "'" to "%27", and "/" to "%2f" respectively in order to properly handle the characters disallowed in XMPP addresses but allowed in sip:/sips: URIs and im:/pres: URIs as shown in Column 3 of Table 3 above (this is consistent with [XEP-0106]).
6. Apply Nodeprep profile of Stringprep [RFC3454] or its replacement (see [RFC6122] and [I-D.ietf-xmpp-6122bis]) for canonicalization (OPTIONAL).
7. Recombine local part with mapped hostname to form a local@domain address (bare JID).
8. If the (SIP) address contained a "gr" URI parameter, append a slash character "/" and the "gr" value to the local@domain address to form a local@domain/resource address (full JID).

5.5. XMPP to SIP

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

1. Split XMPP address into localpart (mapping described in remaining steps), domainpart (hostname; mapping is out of scope), and resourcepart (specifier for particular device or connection, for which an OPTIONAL mapping is described below).
2. Apply Nodeprep profile of [RFC3454] or its replacement (see [RFC6122] and [I-D.ietf-xmpp-6122bis]) for canonicalization of the XMPP localpart (OPTIONAL).
3. Translate "%26" to "&", "%27" to "'", and "%2f" to "/" respectively (this is consistent with [XEP-0106]).
4. Determine if the foreign domain supports im: and pres: URIs (discovered via [RFC2782] lookup as specified in [RFC6121]), else assume that the foreign domain supports sip:/sips: URIs.
5. If converting into im: or pres: URI, for each byte, if the byte is in the set (,.,;[\] (i.e., the partial complement from Row 3, Column 2 of Table 3 above) or is a UTF-8 character outside the ASCII range then percent-encode that byte to "%hexhex" format. If converting into sip: or sips: URI, for each byte, if the byte is in the set #%[\]^`{|} (i.e., the partial complement from Row 3, Column 1 of Table 3 above) or is a UTF-8 character outside the ASCII range then percent-encode that byte to "%hexhex" format.
6. Combine resulting local part with mapped hostname to form local@domain address.
7. Prepend with 'im:' scheme (for XMPP <message/> stanzas) or 'pres:' scheme (for XMPP <presence/> stanzas) if foreign domain supports these, else prepend with 'sip:' or 'sips:' scheme according to local service policy.
8. If the XMPP address included a resourcepart and the destination URI scheme is 'sip:' or 'sips:', optionally append the slash character '/' and then append the resourcepart (making sure to

percent-encode any UTF-8 characters outside the ASCII range).

6. Error Condition Mapping

SIP response codes are specified in [RFC3261] and XMPP error conditions are specified in [RFC6120].

6.1. XMPP to SIP

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

XMPP Error Condition	SIP Response Code
<bad-request/>	400
<conflict/>	400
<feature-not-implemented/>	501
<forbidden/>	403
<gone/>	410
<internal-server-error/>	500
<item-not-found/>	404
<jid-malformed/>	484
<not-acceptable/>	406
<not-allowed/>	405
<not-authorized/>	401
<payment-required/>	402
<recipient-unavailable/>	480
<redirect/>	300
<registration-required/>	407
<remote-server-not-found/>	502
<remote-server-timeout/>	504
<resource-constraint/>	500
<service-unavailable/>	503
<subscription-required/>	407
<undefined-condition/>	400
<unexpected-request/>	491

6.2. SIP to XMPP

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

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

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

7. Security Considerations

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

8. IANA Considerations

This document requests no actions of IANA.

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.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", RFC 4395, February 2006.
- [RFC5627] Rosenberg, J., "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)", RFC 5627, October 2009.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.
- [RFC6122] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 6122, March 2011.
- [UNICODE] The Unicode Consortium, "The Unicode Standard, Version 6.2", 2012, <<http://www.unicode.org/versions/Unicode6.2.0/>>.

9.2. Informative References

- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
- [RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.
- [RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("STRINGPREP")", RFC 3454, December 2002.
- [RFC3856] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.
- [RFC3859] Peterson, J., "Common Profile for Presence (CPP)", RFC 3859, August 2004.
- [RFC3860] Peterson, J., "Common Profile for Instant Messaging (CPIM)", RFC 3860, August 2004.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.
- [RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, March 2011.
- [I-D.ietf-xmpp-6122bis] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", draft-ietf-xmpp-6122bis-05 (work in progress), November 2012.
- [XEP-0106] Saint-Andre, P. and J. Hildebrand, "JID Escaping", XSF XEP 0106, May 2005.

Appendix A. Acknowledgements

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

Authors' Addresses

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

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

Avshalom Houri
IBM
Building 18/D, Kiryat Weizmann Science Park
Rehovot 76123
Israel

Email: avshalom@il.ibm.com

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

Email: jhildebr@cisco.com

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

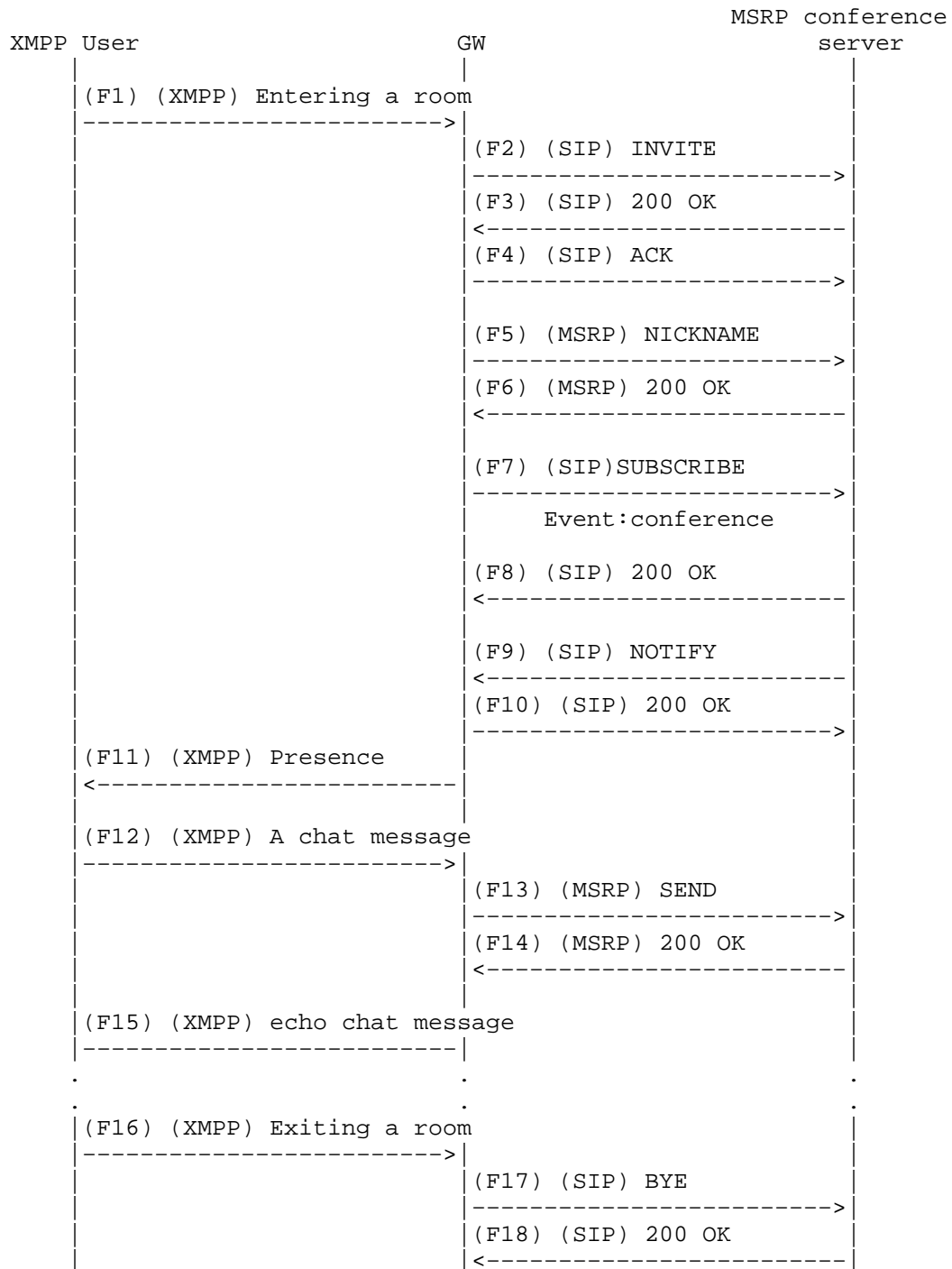
Some text in this document was borrowed from [I-D.saintandre-sip-xmpp-core] and from [XEP-0045].

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.

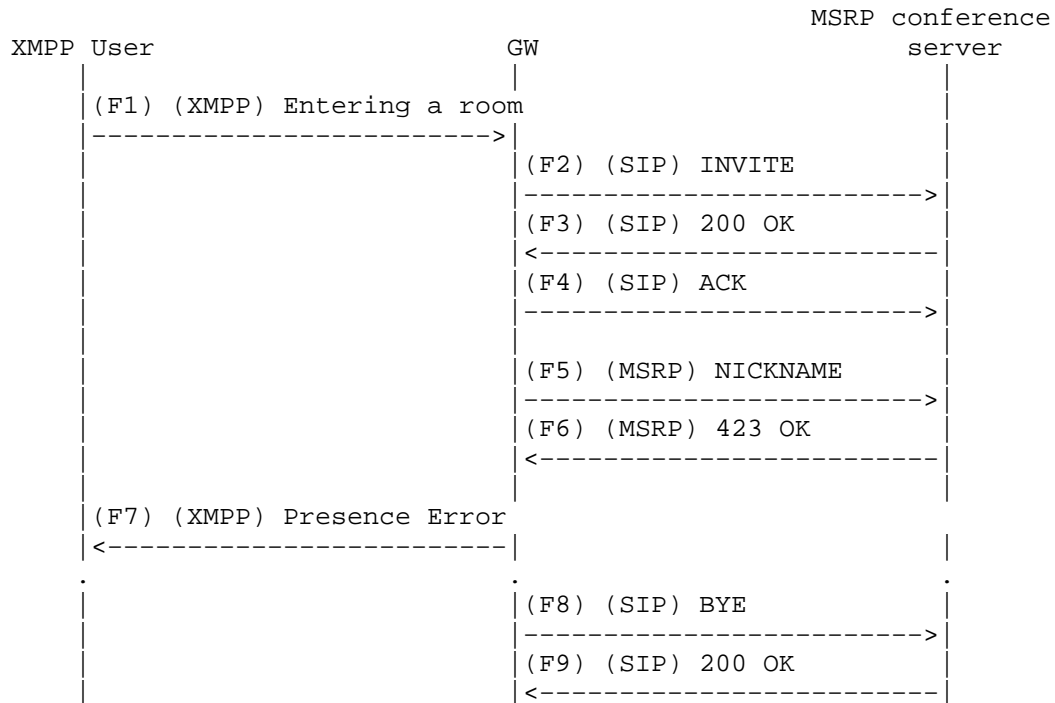
Juliet MAY include a custom exit message in the presence stanza of type "unavailable"

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

```

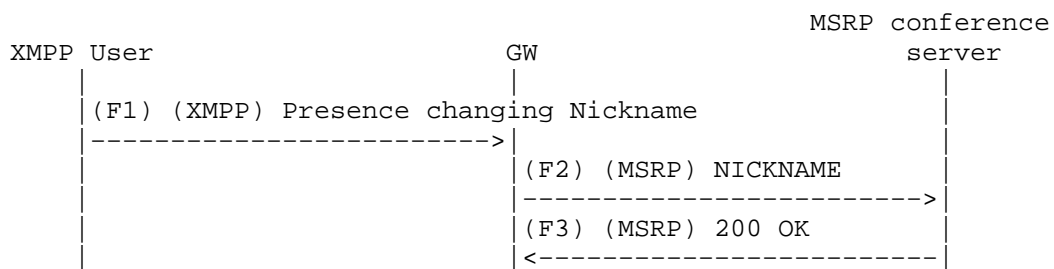
MSRP a786hjs2 423 Nickname usage failed
To-Path: msrp://x2s.example.com:7654/jshA7weztas;tcp
From-Path: msrp://s2x.shakespeare.net:12763/kjhd37s2s20w2a;tcp
-----a786hjs2
  
```

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



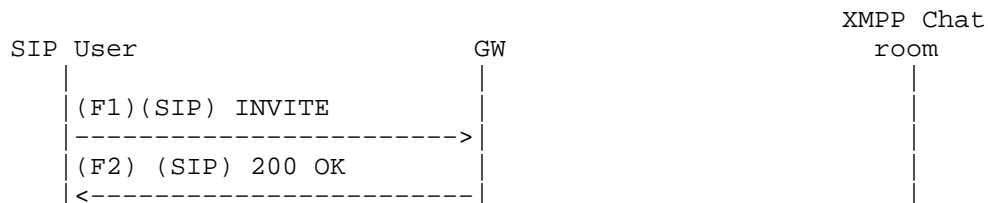
If Juliet decides to changing her nickname within the room, she SHOULD send an update presence information to the room, specifically she SHOULD send a new Nickname in the same room.

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.

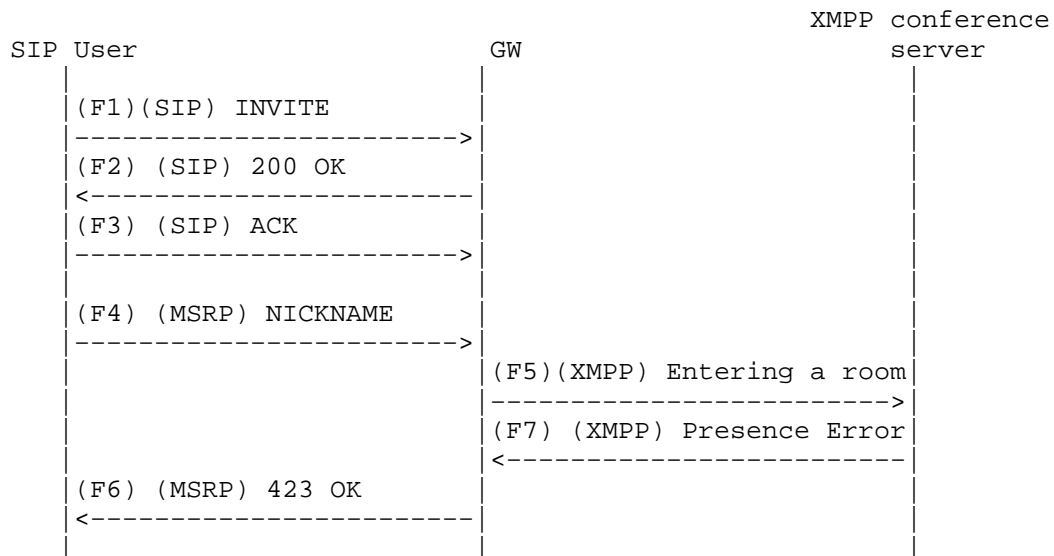
Example: (F19) Juliet exiting a chatroom

```

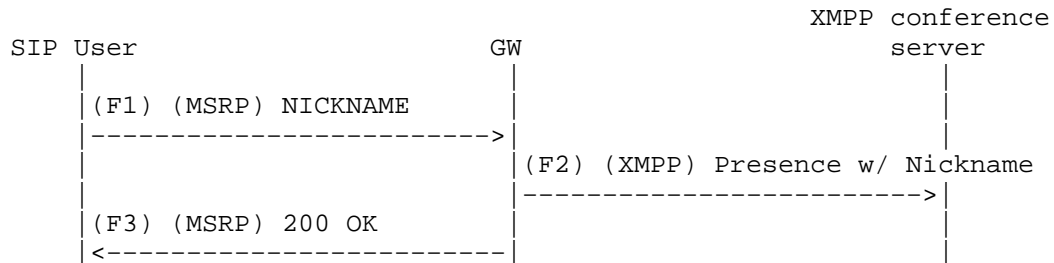
<presence from='romeo@example.com'
          to='verona@chat.shakespeare.net/romeo'
          type='unavailable'>
</presence>

```

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

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

P. Saint-Andre
Cisco Systems, Inc.
A. Hour
IBM
J. Hildebrand
Cisco Systems, Inc.
October 15, 2012

Interworking between the Session Initiation Protocol (SIP) and the
Extensible Messaging and Presence Protocol (XMPP): Instant Messaging
draft-saintandre-sip-xmpp-im-02

Abstract

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

Status of this Memo

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	3
2. Terminology	3
3. Instant Messages	3
3.1. Overview	4
3.2. XMPP to SIP	5
3.3. SIP to XMPP	6
4. Content Types	8
5. Security Considerations	8
6. IANA Considerations	9
7. References	9
7.1. Normative References	9
7.2. Informative References	10
Authors' Addresses	10

1. Introduction

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

- o Various extensions to the Session Initiation Protocol ([RFC3261]) for instant messaging, as developed within the SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) Working Group; the relevant specification for instant messaging is [RFC3428]
- o The Extensible Messaging and Presence Protocol (XMPP), which consists of a formalization of the core XML streaming protocols developed originally by the Jabber open-source community; the relevant specifications are [RFC6120] for the XML streaming layer and [RFC6121] for basic presence and instant messaging extensions

One approach to helping ensure interworking between these protocols is to map each protocol to the abstract semantics described in [RFC3860]; that is the approach taken by [I-D.ietf-simple-cpim-mapping] and [RFC3922]. The approach taken in this document is to directly map semantics from one protocol to another (i.e., from SIP/SIMPLE to XMPP and vice-versa).

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

2. Terminology

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

3. Instant Messages

3.1. Overview

Both XMPP and IM-aware SIP systems enable entities (often but not necessarily human users) to send "instant messages" to other entities. The term "instant message" usually refers to messages sent between two entities for delivery in close to real time (rather than messages that are stored and forwarded to the intended recipient upon request). Generally there are three kinds of instant message:

- o Single messages, which are sent from the sender to the recipient outside the context of any one-to-one chat session or multi-user text conference.
- o Chat messages, which are sent from the sender to the recipient in the context of a "messaging session" between the two entities.
- o Groupchat messages, which are sent from a sender to multiple recipients in the context of a text conference.

This document covers single messages only, since they form the "lowest common denominator" for instant messaging on the Internet. It is likely that future documents will address one-to-one chat sessions and multi-user chat.

Instant messaging using XMPP message stanzas of type "normal" is specified in [RFC6121]. Instant messaging using SIP requests of type MESSAGE (often called "page-mode" messaging) is specified in [RFC3428].

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

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

3.2. XMPP to SIP

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

Example: XMPP user sends message:

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

Upon receiving such a stanza, the XMPP server to which Juliet has connected either delivers it to a local recipient (if the hostname in the 'to' attribute matches one of the hostnames serviced by the XMPP server) or attempts to route it to the foreign domain that services the hostname in the 'to' attribute. Naturally, in this document we assume that the hostname in the 'to' attribute is an IM-aware SIP service hosted by a separate server. As specified in [RFC6121], the XMPP server needs to determine the identity of the foreign domain, which it does by performing one or more DNS SRV lookups [RFC2782]. For message stanzas, the order of lookups recommended by [RFC6121] is to first try the "_xmpp-server" service as specified in [RFC6120] and to then try the "_im" service as specified in [RFC3861]. Here we assume that the first lookup will fail but that the second lookup will succeed and return a resolution "_im._simple.example.net.", since we have already assumed that the example.net hostname is running a SIP instant messaging service. (Note: The XMPP server may have previously determined that the foreign domain is a SIMPLE server, in which case it would not need to perform the SRV lookups; the caching of such information is a matter of implementation and local service policy, and is therefore out of scope for this document.)

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

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

Example: XMPP user sends message (SIP transformation):

```
| MESSAGE sip:romeo@example.net SIP/2.0
| Via: SIP/2.0/TCP x2s.example.com;branch=z9hG4bK776sgdkse
| Max-Forwards: 70
| From: sip:juliet@example.com;tag=49583
| To: sip:romeo@example.net
| Call-ID: Hr0zny9l3@example.com
| CSeq: 1 MESSAGE
| Content-Type: text/plain
| Content-Length: 35
|
| Art thou not Romeo, and a Montague?
```

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

Table 4: Message syntax mapping from XMPP to SIP

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

3.3. SIP to XMPP

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

Example: SIP user sends message:

```
| MESSAGE sip:juliet@example.com SIP/2.0
| Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKeskdg677
| Max-Forwards: 70
| From: sip:romeo@example.net;tag=38594
| To: sip:juliet@example.com
| Call-ID: M4spr4vdu@example.net
| CSeq: 1 MESSAGE
| Content-Type: text/plain
| Content-Length: 44
|
| Neither, fair saint, if either thee dislike.
```

Section 5 of [RFC3428] stipulates that a SIP User Agent presented with an im: URI should resolve it to a sip: or sips: URI. Therefore we assume that the To header of a request received by a SIMPLE-XMPP gateway will contain a sip: or sips: URI. The gateway SHOULD resolve that address to an im: URI for SIP MESSAGE requests, then follow the rules in [RFC3861] regarding the "_im" SRV service for the target domain contained in the To header. If SRV address resolution fails for the "_im" service, the gateway MAY attempt a lookup for the "_xmpp-server" service as specified in [RFC6120] or MAY return an error to the sender (the SIP "502 Bad Gateway" error seems most appropriate; see [I-D.saintandre-sip-xmpp-core] for details). If SRV address resolution succeeds, the gateway is responsible for translating the request into an XMPP message stanza from the SIP user to the XMPP user and returning a SIP "200 OK" message to the sender:

Example: SIP user sends message (XMPP transformation):

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

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

Table 5: Message syntax mapping from SIP to XMPP

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

Note: When transforming SIP page-mode messages, a SIMPLE-XMPP gateway SHOULD specify no XMPP 'type' attribute or a 'type' attribute whose value is "normal" (alternatively, the value of the 'type' attribute MAY be "chat", although it SHOULD NOT be "headline" and MUST NOT be "groupchat").

Note: See the Content Types (Section 4) of this document regarding handling of SIP message bodies that contain content types other than plain text.

4. Content Types

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

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

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

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

5. Security Considerations

Detailed security considerations for instant messaging protocols are

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

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

[RFC2779] requires that conformant technologies shall include methods for blocking communications from unwanted addresses. Such blocking is the responsibility of conformant technology (e.g., XMPP or SIP) and is out of scope for this memo.

6. IANA Considerations

This document requests no actions of IANA.

7. References

7.1. Normative References

- [I-D.saintandre-sip-xmpp-core]
Saint-Andre, P., Houri, A., and J. Hildebrand,
"Interworking between the Session Initiation Protocol
(SIP) and the Extensible Messaging and Presence Protocol
(XMPP): Core", draft-saintandre-sip-xmpp-core-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.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
specifying the location of services (DNS SRV)", RFC 2782,
February 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
A., Peterson, J., Sparks, R., Handley, M., and E.

Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

- [RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.
- [RFC3861] Peterson, J., "Address Resolution for Instant Messaging and Presence", RFC 3861, August 2004.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.
- [RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, March 2011.

7.2. Informative References

- [I-D.ietf-simple-cpim-mapping]
Rosenberg, J. and B. Campbell, "CPIM Mapping of SIMPLE Presence and Instant Messaging",
draft-ietf-simple-cpim-mapping-01 (work in progress),
June 2002.
- [RFC2779] Day, M., Aggarwal, S., and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.
- [RFC3860] Peterson, J., "Common Profile for Instant Messaging (CPIM)", RFC 3860, August 2004.
- [RFC3862] Klyne, G. and D. Atkins, "Common Presence and Instant Messaging (CPIM): Message Format", RFC 3862, August 2004.
- [RFC3922] Saint-Andre, P., "Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM)", RFC 3922, October 2004.
- [XEP-0071]
Saint-Andre, P., "XHTML-IM", XSF XEP 0071, January 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

Avshalom Houri
IBM
Building 18/D, Kiryat Weizmann Science Park
Rehovot 76123
Israel

Email: avshalom@il.ibm.com

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

Email: jhildebr@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 9, 2013

P. Saint-Andre
Cisco Systems, Inc.
A. Hour
IBM
J. Hildebrand
Cisco Systems, Inc.
February 5, 2013

Interworking between the Session Initiation Protocol (SIP) and the
Extensible Messaging and Presence Protocol (XMPP): Presence
draft-saintandre-sip-xmpp-presence-04

Abstract

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

Status of this Memo

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 August 9, 2013.

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	3
2. Terminology	3
3. Discussion Venue	3
4. Presence Subscriptions	4
4.1. Overview	4
4.2. XMPP to SIP	4
4.3. SIP to XMPP	8
5. Presence Notifications	11
5.1. Overview	11
5.2. XMPP to SIP	12
5.3. SIP to XMPP	15
6. Content Types	17
7. Security Considerations	18
8. IANA Considerations	19
9. References	19
9.1. Normative References	19
9.2. Informative References	20
Appendix A. Acknowledgements	21
Authors' Addresses	21

1. Introduction

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

- o Various extensions to the Session Initiation Protocol ([RFC3261]) for instant messaging, as developed within the SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) Working Group; the relevant specification for presence is [RFC3856]
- o The Extensible Messaging and Presence Protocol (XMPP), which consists of a formalization of the core XML streaming protocols developed originally by the Jabber open-source community; the relevant specifications are [RFC6120] for the XML streaming layer and [RFC6121] for basic presence and instant messaging extensions

One approach to helping ensure interworking between these protocols is to map each protocol to the abstract semantics described in [RFC3860]; that is the approach taken by [I-D.ietf-simple-cpim-mapping] and [RFC3922]. The approach taken in this document is to directly map semantics from one protocol to another (i.e., from SIP/SIMPLE to XMPP and vice-versa).

The architectural assumptions underlying such direct mappings are provided in [I-D.saintandre-sip-xmpp-core], including mapping of addresses and error conditions. The mappings specified in this document cover basic presence functionality. Mapping of more advanced functionality is out of scope for this document, but other documents in this series cover such topics.

2. Terminology

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

3. Discussion Venue

The discussion venue for this document is the mailing list of the DISPATCH WG; visit <<https://www.ietf.org/mailman/listinfo/dispatch>> for subscription and archive information.

4. Presence Subscriptions

4.1. Overview

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

As described in [RFC6121], XMPP presence subscriptions are managed using XMPP presence stanzas of type "subscribe", "subscribed", "unsubscribe", and "unsubscribed". The main subscription states are "none" (neither the user nor the contact is subscribed to the other's presence information), "from" (the user has a subscription from the contact), "to" (the user has a subscription to the contact's presence information), and "both" (both user and contact are subscribed to each other's presence information).

As described in [RFC3856], SIP presence subscriptions are managed through the use of SIP SUBSCRIBE events sent from a SIP user agent to an intended recipient who is most generally referenced by an Instant Message URI of the form <pres:user@domain> but who may be referenced by a SIP or SIPS URI of the form <sip:user@domain> or <sips:user@domain>.

The subscription models underlying XMPP and SIP are quite different. For instance, XMPP presence subscriptions are long-lived (indeed permanent if not explicitly cancelled), whereas SIP presence subscriptions are short-lived (the default time to live of a SIP presence subscription is 3600 seconds, as specified in Section 6.4 of [RFC3856]). These differences are addressed below.

4.2. XMPP to SIP

4.2.1. Establishing

An XMPP user initiates a subscription by sending a subscription request to another entity (conventionally called a "contact"), which request the contact either accepts or declines. If the contact accepts the request, the user will have a subscription to the contact's presence information until (1) the user unsubscribes or (2) the contact cancels the subscription. The subscription request is encapsulated in a presence stanza of type "subscribe":

Example: XMPP user subscribes to SIP contact:

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

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

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

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

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

```
| SUBSCRIBE sip:romeo@example.net SIP/2.0  
| Via: SIP/2.0/TCP x2s.example.com;branch=z9hG4bKna998sk  
| From: <sip:juliet@example.com>;tag=ffd2  
| To: <sip:romeo@example.net>  
| Call-ID: 104th3slp@example.com  
| Event: presence  
| Max-Forwards: 70  
| CSeq: 123 SUBSCRIBE  
| Contact: <sip:sipgate.example.com;transport=tcp>  
| Accept: application/pidf+xml  
| Expires: 3600  
| Content-Length: 0
```

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

Example: SIP accepts subscription request:

```
| SIP/2.0 200 OK
| Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
| From: <sip:romeo@example.net>;tag=ffd2
| To: <sip:juliet@example.com>;tag=j89d
| Call-ID: 104th3slp@example.com
| CSeq: 234 SUBSCRIBE
| Contact: <sip:simple.example.net;transport=tcp>
| Expires: 3600
| Content-Length: 0
```

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

Example: SIP user sends presence notification:

```
| NOTIFY sip:192.0.2.1 SIP/2.0
| Via: SIP/2.0/TCP simple.example.net;branch=z9hG4bKna998sk
| From: <sip:romeo@example.net>;tag=yt66
| To: <sip:juliet@example.com>;tag=bi54
| Call-ID: 104th3slp@example.com
| Event: presence
| Subscription-State: active;expires=499
| Max-Forwards: 70
| CSeq: 8775 NOTIFY
| Contact: <sip:simple.example.net;transport=tcp>
| Content-Type: application/pidf+xml
| Content-Length: 193
|
| <?xml version='1.0' encoding='UTF-8'?>
| <presence xmlns='urn:ietf:params:xml:ns:pidf'
|           entity='pres:romeo@example.net'>
|   <tuple id='ID-orchard'>
|     <status>
|       <basic>open</basic>
|       <show xmlns='jabber:client'>away</show>
|     </status>
|   </tuple>
| </presence>
```

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

Example: XMPP user receives acknowledgement from SIP contact:

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

As described under Section 5, the gateway MUST also generate a presence notification to the XMPP user:

Example: XMPP user receives presence notification from SIP contact:

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

4.2.2. Refreshing

It is the responsibility of the XMPP-SIMPLE gateway to set the value of the "Expires" header and to periodically renew the subscription on the SIMPLE side of the gateway so that the subscription appears to be permanent to the XMPP user (e.g., the XMPP-SIMPLE gateway SHOULD send a new SUBSCRIBE request to the SIP user whenever the XMPP user sends initial presence to its XMPP server, i.e., upon initiating a presence session with the XMPP server). See the Security Considerations (Section 7) of this document for important information and requirements regarding the security implications of this functionality.

4.2.3. Cancelling

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

Example: XMPP user unsubscribes from SIP contact:

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

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

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

```
| SUBSCRIBE sip:romeo@example.net SIP/2.0
| Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
| From: <sip:juliet@example.com>;tag=j89d
| To: <sip:romeo@example.net>;tag=xfg9
| Call-ID: 1ckm32@example.com
| Event: presence
| Max-Forwards: 70
| CSeq: 789 SUBSCRIBE
| Contact: <sip:x2s.example.com;transport=tcp>
| Accept: application/pidf+xml
| Expires: 0
| Content-Length: 0
```

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

Example: XMPP user receives unsubscribed notification:

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

4.3. SIP to XMPP

4.3.1. Establishing

A SIP user initiates a subscription to a contact's presence information by sending a SIP SUBSCRIBE request to the contact. The following is an example of such a request:

Example: SIP user subscribes to XMPP contact:

```
| SUBSCRIBE sip:juliet@example.com SIP/2.0
| Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk
| From: <sip:romeo@example.net>;tag=xfg9
| To: <sip:juliet@example.com>;tag=ur93
| Call-ID: 4wcm0n@example.net
| Event: presence
| Max-Forwards: 70
| CSeq: 263 SUBSCRIBE
| Contact: <sip:simple.example.net;transport=tcp>
| Accept: application/pidf+xml
| Content-Length: 0
```

Notice that the "Expires" header was not included in the SUBSCRIBE

request; this means that the default value of 3600 (i.e., 3600 seconds = 1 hour) applies.

Upon receiving such a request, a SIMPLE-XMPP gateway is responsible for translating it into an XMPP subscription request from the SIP user to the XMPP user:

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

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

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

4.3.2. Refreshing

It is the responsibility of the SIMPLE-XMPP gateway to properly handle the difference between short-lived SIP presence subscriptions and long-lived XMPP presence subscriptions. The gateway has two options when the SIP user's subscription expires:

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

Which of these options the SIMPLE-XMPP gateway chooses is up to the implementation.

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

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

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

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

Example: SIP subscription expires (treated as long-lived by gateway):

```
| NOTIFY sip:192.0.2.2 SIP/2.0  
| Via: SIP/2.0/TCP s2x.example.net;branch=z9hG4bKna998sk  
| From: <sip:juliet@example.com>;tag=ur93  
| To: <sip:romeo@example.net>;tag=pq72  
| Call-ID: j4s0h4vny@example.com  
| Event: presence  
| Subscription-State: terminated;reason=timeout  
| Max-Forwards: 70  
| CSeq: 232 NOTIFY  
| Contact: <sip:sipgate.example.com;transport=tcp>  
| Content-Type: application/pidf+xml  
| Content-Length: 194  
  
| <?xml version='1.0' encoding='UTF-8'?>  
| <presence xmlns='urn:ietf:params:xml:ns:pidf'  
|           entity='pres:juliet@example.com'  
|           <tuple id='ID-balcony'  
|               <status>  
|                 <basic>closed</basic>  
|               </status>  
|             </tuple>  
| </presence>
```

Example: SIP subscription expires (treated as long-lived by gateway):

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

4.3.3. Cancelling

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

Example: SIP user cancels subscription:

```
| SUBSCRIBE sip:192.0.2.1 SIP/2.0
| Via: SIP/2.0/TCP simple.example.net;branch=z9hG4bKna998sk
| From: <sip:romeo@example.net>;tag=yt66
| To: <sip:juliet@example.com>;tag=bi54
| Call-ID: ltsnlce@example.net
| Event: presence
| Max-Forwards: 70
| CSeq: 8775 SUBSCRIBE
| Contact: <sip:simple.example.net;transport=tcp>
| Expires: 0
| Content-Length: 0
```

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

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

5. Presence Notifications

5.1. Overview

Both XMPP and presence-aware SIP systems enable entities (often but not necessarily human users) to send presence notifications to other entities. At a minimum, the term "presence" refers to information about an entity's availability for communication on a network (on/off), often supplemented by information that further specifies the entity's communications context (e.g., "do not disturb"). Some systems and protocols extend this notion even further and refer to any relatively ephemeral information about an entity as a kind of presence; categories of such "extended presence" include geographical location (e.g., GPS coordinates), user mood (e.g., grumpy), user activity (e.g., walking), and ambient environment (e.g., noisy). In this document, we focus on the "least common denominator" of network availability only, although future documents may address broader notions of presence, including extended presence.

[RFC6121] defines how XMPP presence stanzas can indicate availability (via absence of a 'type' attribute) or lack of availability (via a 'type' attribute with a value of "unavailable"). SIP presence using a SIP event package for presence is specified in [RFC3856].

As described in [RFC6121], presence information about an entity is communicated by means of an XML <presence/> stanza sent over an XML stream. In this document we will assume that such a presence stanza is sent from an XMPP client to an XMPP server over an XML stream negotiated between the client and the server, and that the client is controlled by a human user (again, this is a simplifying assumption introduced for explanatory purposes only). In general, XMPP presence is sent by the user to the user's server and then broadcasted to all entities who are subscribed to the user's presence information.

As described in [RFC3856], presence information about an entity is communicated by means of a SIP NOTIFY event sent from a SIP user agent to an intended recipient who is most generally referenced by an Instant Message URI of the form <pres:user@domain> but who may be referenced by a SIP or SIPS URI of the form <sip:user@domain> or <sips:user@domain>. Here again we introduce the simplifying assumption that the user agent is controlled by a human user.

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

5.2. XMPP to SIP

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

Example: XMPP user sends presence notification:

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

Upon receiving such a stanza, the XMPP server to which Juliet has connected broadcasts it to all subscribers who are authorized to receive presence notifications from Juliet (this is similar to the SIP NOTIFY method). For each subscriber, broadcasting the presence notification involves either delivering it to a local recipient (if the hostname in the subscriber's address matches one of the hostnames serviced by the XMPP server) or attempting to route it to the foreign domain that services the hostname in the subscriber's address.

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

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

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

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

```

NOTIFY sip:192.0.2.2 SIP/2.0
Via: SIP/2.0/TCP x2s.example.com;branch=z9hG4bKna998sk
From: <sip:juliet@example.com>;tag=gh19
To: <sip:romeo@example.net>;tag=yt66
Call-ID: j4s0h4vny@example.com
Event: presence
Subscription-State: active;expires=599
Max-Forwards: 70
CSeq: 157 NOTIFY
Contact: <sip:sipgate.example.com;transport=tcp>
Content-Type: application/pidf+xml
Content-Length: 192

<?xml version='1.0' encoding='UTF-8'?>
<presence xmlns='urn:ietf:params:xml:ns:pidf'
  entity='pres:juliet@example.com'>
  <tuple id='ID-balcony'>
    <status>
      <basic>open</basic>
      <show xmlns='jabber:client'>away</show>
    </status>
  </tuple>
</presence>

```

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

Table 6: Presence syntax mapping from XMPP to SIP

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

Note the following regarding these mappings:

1. Only a presence stanza that lacks a 'type' attribute or whose 'type' attribute has a value of "unavailable" should be mapped by an XMPP-SIMPLE gateway to a SIP NOTIFY request, since those are the only presence stanzas that represent notifications.
2. The PIDF schema defines the tuple 'id' attribute as having a datatype of "xs:ID"; because this datatype is more restrictive than the "xs:string" datatype for XMPP resourceparts (in particular, a number is not allowed as the first character of an ID), it is RECOMMENDED to prepend the resourcepart with "ID-" or some other alphabetic string when mapping from XMPP to SIP.
3. Because the lack of a 'type' attribute indicates that an XMPP entity is available for communications, the gateway SHOULD map that information to a PIDF <basic/> status of "open". Because a 'type' attribute with a value of "unavailable" indicates that an XMPP entity is not available for communications, the gateway SHOULD map that information to a PIDF <basic/> status of "closed".
4. When the XMPP-SIMPLE gateway receives XMPP presence of type "unavailable" from the XMPP contact, it SHOULD (1) send a SIP NOTIFY request to the SIP user containing a PIDF document specifying that the XMPP contact now has a basic status of "closed" and (2) send a SIP SUBSCRIBE request to the SIP user with an "Expires" header set to a value of "0" (zero).
5. Some implementations support custom extensions to encapsulate this information; however, there is no need to standardize a PIDF extension for this purpose, since PIDF is already extensible and thus the <show/> element can be included directly, qualified by the 'jabber:client' namespace in the PIDF XML. The examples in this document illustrate this usage.
6. The XML character data of the XMPP <status/> element MAY be mapped to the character data of the PIDF <note/> element.

5.3. SIP to XMPP

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

Example: SIP user sends presence notification:

```
| NOTIFY sip:192.0.2.1 SIP/2.0
| Via: SIP/2.0/TCP simple.example.net;branch=z9hG4bKna998sk
| From: <sip:romeo@example.net>;tag=yt66
| To: <sip:juliet@example.com>;tag=bi54
| Call-ID: j0sj4svlm@example.net
| Event: presence
| Subscription-State: active;expires=499
| Max-Forwards: 70
| CSeq: 8775 NOTIFY
| Contact: <sip:simple.example.net;transport=tcp>
| Content-Type: application/pidf+xml
| Content-Length: 193
|
| <?xml version='1.0' encoding='UTF-8'?>
| <presence xmlns='urn:ietf:params:xml:ns:pidf'
|   entity='pres:romeo@example.net'>
|   <tuple id='orchard'>
|     <status>
|       <basic>closed</basic>
|     </status>
|   </tuple>
| </presence>
```

Upon receiving such a request, a SIMPLE-XMPP gateway is responsible for translating it into an XMPP presence stanza from the SIP user to the XMPP user:

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

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

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

Table 7: Presence syntax mapping from SIP to XMPP

SIP Header or PIDF Data	XMPP Element or Attribute
basic status	type [1]
Content-Language	xml:lang
CSeq	id (OPTIONAL)
From	from
priority for tuple	<priority/>
To	to
body of MESSAGE	<body/>

Note the following regarding these mappings:

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

6. Content Types

SIP requests of type NOTIFY normally contain presence information encapsulated using the "application/pidf+xml" content type. The recommended procedures for SIMPLE-to-XMPP gateways to use in handling these content types are as follows.

The "application/pidf+xml" content type is specified in [RFC3863]. The Presence Information Data Format defines a common data format for presence protocols that conform to the Common Profile for Presence ([RFC3859]), enabling presence information to be transferred across CPP-compliant protocol boundaries without modification, with attendant benefits for end-to-end encryption and performance. Because the syntax for the "application/pidf+xml" content type is Extensible Markup Language ([XML]), it is straightforward to send PIDF data over the Extensible Messaging and Presence Protocol (XMPP) ([RFC6120]), since XMPP is simply an XML streaming protocol.

In addition to following the syntax mappings specified in Section 5, a SIMPLE-to-XMPP gateway MAY encapsulate PIDF data within an "extended namespace" contained in an XMPP presence stanza. The RECOMMENDED method is to include the PIDF <presence/> element as a child of the XMPP <presence/> stanza. Although it may appear that this would be potentially confusing, the inclusion of the 'urn:ietf:params:xml:ns:pidf' namespace ensures that PIDF data is kept separate from XMPP presence data (in accordance with [XML-NAMES]). The following is a simple example of encapsulating

PIDF data within an "extended namespace" in XMPP:

A basic example of PIDF over XMPP:

```
<presence from='romeo@example.net/orchard'
  to='nurse@example.com'
  xml:lang='en'>
  <status>Wooin Juliet</status>
  <presence xmlns='urn:ietf:params:xml:ns:pidf'
    entity='pres:romeo@example.net'>
    <tuple id='orchard'>
      <status>
        <basic>open</basic>
        <show xmlns='jabber:client'>dnd</show>
        <note>Wooin Juliet</note>
      </status>
    </tuple>
  </presence>
</presence>
```

7. Security Considerations

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

The mismatch between long-lived XMPP presence subscriptions and short-lived SIP presence subscriptions introduces the possibility of an amplification attack launched from the XMPP network against a SIP presence server. To help prevent such an attack, access to an XMPP-SIMPLE gateway that is hosted on the XMPP network SHOULD be restricted to XMPP users associated with a single domain or trust realm (e.g., a gateway hosted at simple.example.com should allow only users within the example.com domain to access the gateway, not users within example.org, example.net, or any other domain); if a SIP presence server receives communications through an XMPP-SIMPLE gateway from users who are not associated with a domain that is so related to the hostname of the gateway, it MAY (based on local service provisioning) refuse to service such users or refuse to communicate with the gateway. Furthermore, whenever an XMPP-SIMPLE gateway seeks to refresh an XMPP user's long-lived subscription to a SIP user's presence, it MUST first send an XMPP <presence/> stanza of type "probe" from the address of the gateway to the "bare JID" (user@domain.tld) of the XMPP user, to which the user's XMPP server MUST respond in accordance with [RFC6121]; however, the administrator of an XMPP-SIMPLE gateway MAY (based on local service provisioning) exempt "known good" XMPP servers from this check (e.g., the XMPP

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

8. IANA Considerations

This document requests no actions of IANA.

9. References

9.1. Normative References

- [I-D.saintandre-sip-xmpp-core] Saint-Andre, P., Houri, A., and J. Hildebrand, "Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Core", draft-saintandre-sip-xmpp-core-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.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3856] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.
- [RFC3861] Peterson, J., "Address Resolution for Instant Messaging and Presence", RFC 3861, August 2004.
- [RFC3863] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, August 2004.
- [RFC6665] Roach, A., "SIP-Specific Event Notification", RFC 6665, July 2012.
- [XML] Paoli, J., Maler, E., Sperberg-McQueen, C., Yergeau, F., and T. Bray, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", World Wide Web Consortium Recommendation REC-xml-20060816, August 2006,

<<http://www.w3.org/TR/2006/REC-xml-20060816>>.

[XML-NAMES]

Bray, T., Hollander, D., and A. Layman, "Namespaces in XML", W3C REC-xml-names, January 1999, <<http://www.w3.org/TR/REC-xml-names>>.

[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.ietf-simple-cpim-mapping]

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

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

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

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

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

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

[XEP-0107]

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

[XEP-0108]

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

Appendix A. Acknowledgements

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

Authors' Addresses

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

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

Avshalom Houri
IBM
Building 18/D, Kiryat Weizmann Science Park
Rehovot 76123
Israel

Email: avshalom@il.ibm.com

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

Email: jhildebr@cisco.com

